ACTIVIST-DRIVEN INNOVATION: UMA HISTÓRIA INTERPRETATIVA DO *SOFTWARE* LIVRE

Maria Caramez Carlotto
Pablo Ortellado

Introdução

Os estudos sobre a produção de conhecimento – sejam eles da área de sociologia, economia, administração ou mesmo filosofia – reconhecem, em geral, a validade da ideia de que existem diferentes *regimes ou sistemas de produção do conhecimento* – aos quais correspondem formas específicas de estrutura organizacional, organização do trabalho, regime de recompensa, motivação subjetiva, práticas e valores sociais, e formas de gestão da propriedade intelectual (Merton, 1957, 1963, 1973; Bourdieu, 2004; Biaglioli, 1998; Nelson, 2004; Shinn, 2000a e b, 2008b).

Existem muitas possibilidades de classificação dos regimes de produção de conhecimento, dependendo da ênfase que se dê às diferentes dimensões que os caracterizam. Assim, a primeira hipótese do presente trabalho é a de que existem dois regimes sob os quais se organizou, *historicamente*, a atividade de desenvolvimento de *software*: um *regime público/científico* e um *regime privado/empresarial*.[1] Esta compreensão – de que existem dois regimes distintos para a produção de *software* – é cada vez mais comum na literatura sobre o tema (Gambardella e Hall, 2006; Laat, 2005; Bonaccorsi e Rossi, 2005, Osterloch e Rota, 2007). O que não é tão comum, e se configura, portanto, como a contribuição mais original deste artigo é, de um lado, a abordagem histórica da configuração desses dois regimes e, de outro, a análise dos fatores que determi-

* Uma versão preliminar deste texto foi apresentada nas VII Jornadas Latinoamericanas de Estudios Sociales de la Ciencia y la Tecnología no Rio de Janeiro em junho de 2008. Agradecemos aos importantes comentários feitos por Hernán Thomas na ocasião, que nos ajudaram a corrigir erros e imprecisões. Os autores são, no entanto, os únicos responsáveis pelas falhas no texto.

Artigo recebido em agosto/2010
Aprovado em fevereiro/2011

nam o sucesso técnico e "comercial" de um regime em detrimento do outro.

Assim, trabalhamos com duas hipóteses complementares: a primeira, de que o desenvolvimento do *software* livre pertence *historicamente* ao regime público/científico de produção do conhecimento, ou seja, o *software* livre mimetiza a organização da "comunidade" científica porque tem nela as suas raízes históricas; e a segunda, de que em uma "competição de mercado" o regime público/científico se mostrou mais eficiente e, por isso, obrigou as empresas que trabalhavam no regime privado/empresarial a adotar o *software* livre ou de código aberto.

A colaboração pré-competitiva

O PACT I e II

Há um consenso mais ou menos difundido entre os principais estudiosos do *software* livre e/ou de código aberto de que as práticas colaborativas para o desenvolvimento de *softwares* são anteriores ao projeto GNU e à criação da Free Software Foundation (Raymond, 2000; Weber, 2004; Kim, 2005). Segundo Raymond:

> [...] a FSF [Free Software Foundation] nunca foi a única iniciativa na área. Sempre existiu uma força mais silenciosa, menos confrontadora e mais amigável em relação ao mercado na cultura *hacker*. Os "pragmáticos" foram leais menos a uma ideologia do que a um conjunto de tradições de engenharia fundado nos primeiros esforços do código aberto que precederam a FSF. Essas tradições incluem, mais notadamente, as culturas interligadas do Unix e da Internet pré-comercial (2000, p. 2).

De fato, não é difícil perceber – analisando a história das práticas de desenvolvimento de *software* – que a colaboração parece ser tão antiga quanto o próprio *software* em si. Quando, no começo da década de 1950, surgiram os primeiros computadores comerciais – por exemplo, o 701 da IBM lançado em 1952 a um custo de manutenção mensal de U$ 15,000 –, a necessidade de desenvolver programas que os fizessem funcionar era tão grande quanto a dificuldade técnica envolvida nessa atividade. Também era um fator importante a escassez de mão de obra capaz de levar a cabo a tarefa[2] e a inexistência de um mercado bem desenvolvido para esse tipo de produto. Neste contexto – de imensa necessidade de um rápido processo de desenvolvimento tecnológico e de um alto custo e risco envolvido na sua realização –, o processo de escrita das primeiras linhas de código acabou fortalecendo um ambiente favorável às práticas de colaboração.

O acontecimento que marcou, historicamente, a consolidação de práticas colaborativas nos processos de desenvolvimento de *software* foi a criação do Project For Advancement of Coding Techniques (PACT) em 1952. O PACT envolvia diversas empresas da área de informática[3] num esforço cooperativo para o desenvolvimento de um sistema operacional[4] automático para o IBM 701. O projeto teve duas edições – os assim chamados PACT-I e PACT-II – cujo resultado final foi o primeiro *software* desenvolvido de forma colaborativa por empregados de firmas diferentes (Bernstein, 1973), um esforço denominado, modernamente, de pesquisa pré-competitiva.

Por ser o primeiro projeto colaborativo de maior relevância que se conhece, o PACT é considerado, por alguns autores, um marco fundamental no surgimento de uma "cultura da colaboração" entre programadores de *software*. Para Eric E. Kim, por exemplo, é possível dizer que antes do PACT a cultura da colaboração, condição necessária para o surgimento de projetos colaborativos, simplesmente não existia (Kim, 2005).[5]

A nossa hipótese, neste artigo, é um pouco diferente da de autores como Kim e outros que atribuem o surgimento da cultura de colaboração entre programadores de *software* a projetos específicos de sistemas operacionais como o PACT ou mesmo ao BSD/Unix. Da nossa perspectiva, a cultura da colaboração está ligada, antes, à tradição acadêmico/científica na qual esses programadores se formaram. Outros autores (Raymond, 1999; Bonaccorsi e Rossi, 2003; DiBona *et al.*, 1999) também ressaltam as origens acadêmico-científicas da comunidade *hacker*:

> A comunidade de programadores é muito heterogênea, mas a explicação para o enorme resultado alcançado pelo projeto do código aberto reside na cultura *hacker* dos "programadores reais" vindos dos campos da física e das engenharias no pós-guerra. [...] muitas gerações de cientistas da computação foram formados no mundo acadêmico e nos centros de pesquisa das grandes corporações de *software* (Bonaccorsi e Rossi, 2003, p. 1245).

Mas independentemente do fato de o PACT não ter "originado" a cultura de colaboração entre programadores de *software*, o projeto acabou se tornando um marco do sucesso dessa forma colaborativa de produção de *software*, de tal modo que ela passou a ser considerada a "forma ideal" da atividade de desenvolvimento e programação. Segundo Paul Armer, um dos programadores envolvidos no PACT-I:

> O espírito de cooperação entre as organizações participantes e seus representantes durante a formulação do PACT-I foi um dos recursos mais valiosos que surgiu do projeto. É essencial que esse espírito de cooperação continue nos planos de projetos futuros (Armer, 1973, s/p).

Embora acreditemos que o PACT não seja a raiz principal da colaboração incorporada pelo *software* livre, ele certamente contribuiu para consolidar a cultura de colaboração que tinha origem nas práticas científicas. No campo do *software*, esse *ethos* colaborativo se expressou de maneira mais clara em duas experiências: o projeto BSD, que se desenvolveu em torno da Universidade da Califórnia, e o projeto GNU, a partir do MIT. Na experiência californiana, a ausência de uma política de licenciamento de direitos autorais levou a uma incorporação pelo mercado que introduziu práticas competitivas e fragmentou o código; já no projeto GNU, a lição do BSD foi incorporada e uma inovadora estratégia de licenciamento foi desenvolvida para conter a dispersão competitiva e garantir a sustentabilidade da colaboração.

O projeto BSD/Unix

A possibilidade de que os projetos para o desenvolvimento de *softwares* permanecessem no âmbito dos esforços colaborativos – ou, dito de outro modo, no âmbito do regime público/científico de produção do conhecimento – foi assegurada, alguns anos mais tarde, pelo acordo firmado entre o governo dos Estados Unidos e as grandes empresas da área de telefonia da época: AT&T e Western Eletric Company Inc.

Desde janeiro de 1949, o governo norte-americano, através da Divisão Antitruste do Departamento de Justiça, movia uma ação contra a AT&T e a Western Eletric Company Inc. pela violação da Lei Sherman – o famoso dispositivo norte-americano antitrustes e contra a propriedade cruzada de 1890. O processo, que durou mais de sete anos, terminou em 1956, com a deliberação do juiz Thomas F. Meaney, por meio de um *consent decree*,[6] segundo o qual as empresas envolvidas se comprometiam a não desenvolver atividades comerciais fora da área de telegrama e telefonia: "A acusada AT&T é forçada e impedida de se engajar, direta ou indiretamente (por intermédio de suas subsidiárias outras que não a Western e suas subsidiárias) em qualquer negócio que não seja o fornecimento de serviços de comunicação" (United States District Court, 1956, s/p). Mais do que isso, a deliberação de Meaney, e a interpretação que dela tiveram os advogados das duas empresas envolvidas, implicou uma determinação de que todas as patentes da AT&T, da Western Eletric e de suas subsidiárias fossem licenciadas a custos nominais e sem maiores restrições (Salus, 1994, p. 57). Como os Bell Telephone Labs, mais conhecidos como Bell Labs, eram, nessa época, subsidiários tanto da AT&T como da Western Eletric – cada uma possuía 50% dos laboratórios –, eles também ficaram obrigados a licenciar todas as suas patentes considerando as determinações do *consent decree:* ou seja, a um custo mínimo e sem restrições significativas.

Segundo Salus (1994), o *Consent Decree* de 1956 e a interpretação conservadora dada a ele pelos advogados da AT&T e da Western Eletric, tiveram, juntos, um efeito decisivo para a política relativa às pesquisas que os Bell Labs viriam a

desenvolver na área de *software* e programação em geral. Por não serem passíveis de comercialização – dado que eram consideradas, na época, externas às áreas de telegrama e telefonia, às quais deveriam se restringir as operações da AT&T e da Western Eletric –, essas pesquisas puderam se desenvolver, nos laboratórios, com certa liberdade e forte caráter colaborativo.

Assim, em 1964, os Bell Labs começaram a desenvolver um sistema operacional chamado Multics (Multiplexed Information and Computing Service) em parceria com o MIT e com a General Eletric, dentro desse registro colaborativo que marcava as pesquisas do laboratório na área. No entanto, em 1969, depois de cinco anos de pesquisas, o projeto foi abandonado pelos Bell Labs por razões consideradas, na época, de ordem "técnica". A decisão de abandonar o Multics, no entanto, foi antes da direção dos laboratórios do que dos pesquisadores envolvidos no projeto. Entre esses, Ken Thompson e Dennis Ritchie, em especial, estavam convencidos das potencialidades do Multics e se empenharam – a princípio, fora do âmbito da empresa[7] – no projeto de um sistema operacional inspirado no Multics (Weber, 2004).

Em outubro de 1973, na quarta edição do Symposium on Operating Systems Principles,[8] ocorrido na Purdue University de Nova York, Thompson e Ritchie inscreveram um artigo que apresentava para a comunidade de programadores e pesquisadores de *software*, as possibilidades de "uso e implementação" do Unix, um "sistema operacional de propósito geral, multiusuário e interativo para o DEC/PDP –11/40" (Thompson e Ritchie, 1973, p. 1).

O fato de o Unix ter sido apresentado originalmente em um congresso acadêmico mostra o quanto os Bell Labs, apesar de serem um conjunto privado de laboratórios, valorizavam a relação com a academia, sobretudo a norte-americana. Essa valorização do mundo acadêmico refletia-se claramente na sua própria organização interna, que mimetizava, em muitos aspectos, a organização acadêmica de produção do conhecimento.

O Unix, apresentado no Symposium on Operating Systems Principles em 1973, despertou a atenção dos pesquisadores universitários imediatamente, de modo que, já em 1975, mais de quarenta universidades norte-americanas possuíam computadores que rodavam Unix (Salus, 1994). A rápida difusão do Unix foi possibilitada, em grande medida, pela política de licenciamento de *software* da AT&T. Como vimos, uma vez que a empresa não podia explorar comercialmente "*softwares* para computador", nem aferir *royalties* do seu patenteamento por conta das limitações instituídas pelo *Consent Decree* de 1956, a solução foi licenciar *softwares* como o Unix apenas para fins de pesquisa, a custos nominais, mas sem nenhum serviço de suporte ou manutenção, ou seja, sem ônus para a empresa. Essa política de licenciamento teve um efeito duplo e complementar: por conta do seu baixo custo, o Unix disseminou-se rapidamente entre instituições de pesquisa e universidades. Paralelamente, a ausência de qualquer serviço de suporte ou manutenção obrigou a comunidade de usuários de Unix a compartilhar informações, soluções de problemas e atualizações do sistema.

Entre os usuários mais interessados no sistema, estava o pesquisador Robert Fabry do Departamento de Ciência da Computação, Estatística e Matemática da Universidade de Berkeley, Califórnia. Fabry assistiu à apresentação de Ritchie e Thompson no simpósio de Nova York, em 1973, e passou a acompanhar o trabalho deles para fins de pesquisa. O interesse científico de Berkeley e o desinteresse comercial dos Bell Labs no Unix possibilitaram um longo processo de colaboração entre as duas instituições.

Esse processo intensificou-se a partir de 1978, quando os pesquisadores de Berkeley colocam, em um mesmo "pacote de distribuição", o *kernel*[9] do Unix e outros *softwares*, formatando uma linha de distribuição do Unix que ficou conhecida como BSD, ou Berkeley Software Distribution. No ano seguinte, Robert Fabry propôs oficialmente ao Darpa – Defense Advanced Research Projects Agency – um projeto de desenvolvimento de uma versão do BSD – com o *kernel* do Unix – para a Internet. O projeto foi aprovado em 1980, originando o Computer System Research Group, ligado ao Departamento de Ciência da Computação de Berkeley. O grupo desenvolveu, nos dezoito meses previstos para o projeto, uma versão do BSD integrada à

Internet, também conhecida como a 4ª versão da BSD/Unix.

Mas esse processo de relativa liberdade e intensa colaboração inter-institucional passou a enfrentar algumas dificuldades a partir do final dos anos de 1970, quando a incipiente popularização do computador pessoal possibilitou o surgimento de um *mercado* de *software*, levando as empresas a iniciarem políticas mais agressivas de proteção e distribuição dos seus produtos. O documento paradigmático dessa mudança de atitude das empresas é a carta do jovem fundador da Microsoft aos "entusiastas" não profissionais que animavam o hoje lendário Homebrew Computer Club, um fórum de "amadores" interessados em microcomputadores e que se reunia na Universidade de Stanford, entre 1975 e 1977, para compartilhar diferentes versões de programas de computador. A carta, assinada por Bill Gates, reclamava do compartilhamento de um *software*, o Altair-Basic, que a Microsoft havia desenvolvido para o primeiro microcomputador pessoal, o Altair 8800:

> Como a maioria dos amadores deve saber, a maior parte de vocês rouba os seus *softwares*. O *hardware* precisa ser comprado, mas [para vocês] o *software* deveria ser compartilhado. Quem se importa se as pessoas que trabalharam nele serão pagas? Isso é justo? [...] O que vocês fazem é impedir que *software* de qualidade seja escrito. Quem pode bancar fazer trabalho profissional por nada? Que amador pode despender três anos-trabalho em programação, encontrar todos os *bugs*, documentar o trabalho e distribuí-lo de graça? O fato é que ninguém, com a exceção de nós, investiu tanto dinheiro no *software* para amadores. Nós escrevemos o Basic 6800 e estamos escrevendo o APL 8080 e o APL 6800, mas há muito pouco incentivo para tornar esse *software* disponível para os amadores. Falando francamente, o que vocês estão fazendo é roubo (Gates, 1976, p. 2)

Essa mudança na forma como o compartilhamento de *software* passou a ser visto a partir do surgimento de um mercado específico de programas de computador ecoou também na AT&T. A empresa, que desde 1974 estava sendo novamente processada pelo governo norte-americano por monopólio, foi obrigada, em 1982, a se separar tanto dos Bell Labs como da Western Electric. Essas mudanças de natureza política e econômica fizeram com que, a partir de 1983, a AT&T internalizasse o desenvolvimento do Unix, criando o Unix System Labs e alterando radicalmente a forma de distribuição e licenciamento do *software*. Por conta disso, em 1988, a licença do Unix já custava aproximadamente U$ 100 mil chegando, alguns anos depois, a U$250 mil (Weber, 2004, p. 39). A mudança da política de licenciamento do Unix resultou no paulatino arrefecimento da colaboração entre os Bell Labs e a Universidade de Berkeley, que culminou no processo judicial movido pela AT&T contra a universidade californiana em 1992.

O enfraquecimento das redes de colaboração entre Berkeley – que tinha a sua distribuição própria do Unix, a BSD – e a AT&T – que também mantinha a sua versão de distribuição – começou, no entanto, antes mesmo da mudança da política de licenciamento da empresa. Já em 1981, a AT&T proibiu a Universidade de lançar a versão 4.1 da Berkeley Software Distribution (BSD) porque pretendia, ainda naquele ano, lançar a sua distribuição comercialmente.

Assim, a história do Unix na década de 1980 é marcada, de um lado, pela ruptura das relações colaborativas que pautaram o seu desenvolvimento em um contexto em que o mercado de *software* era pouco desenvolvido e que uma das principais empresas envolvidas nesse processo não podia explorar os resultados dessas pesquisas comercialmente; de outro, pela completa fragmentação do processo de desenvolvimento do Unix.

A história da fragmentação do projeto Unix e o seu consequente fracasso na década de 1980, bem como a ascensão do projeto GNU – cuja sigla significa, justamente, "GNU Não é Unix" – nos anos 1990 são processos incompreensíveis sem uma análise da política de licenciamento da Berkeley University. A licença BSD é considerada uma das mais liberais que existem porque permite, *grosso modo*, qualquer tipo de utilização sem imposição de nenhuma exigência específica sobre o caráter dos

futuros licenciamentos derivados.[10] Os termos da licença até 1994 diziam o seguinte:

> A redistribuição e o uso em código-fonte ou binário, com ou sem modificação, é permitida desde que as seguintes condições sejam seguidas: 1. Redistribuições do código-fonte devem conter a nota de *copyright* acima, esta lista de condições e as restrições que se seguem; 2. Redistribuições em forma binária devem reproduzir a nota de *copyright* acima, esta lista de condições e as restrições que se seguem na sua documentação e/ou em outro material oferecido com a distribuição; 3. Todo material de propaganda que mencione vantagens ou uso desse *software* deve conter o seguinte agradecimento: Este produto inclui *software* desenvolvido pela Universidade da Califórnia, Berkeley, e seus contribuidores. 4. Nem o nome da Universidade nem os seus contribuidores podem ser usados para endossar ou promover produtos derivados desse *software* sem a permissão prévia e específica (BSD License, s/d).[11]

Essa forma específica de licenciamento – sem qualquer exigência ou prescrição quanto à forma de distribuição dos softwares derivados – pode ser interpretada, grosso modo, como uma não-licença. Na medida em que abre possibilidade para que versões e modificações feitas em um *software* sejam posteriormente distribuídas sob uma licença restritiva, ou seja, sem a obrigação de publicização do código-fonte, a licença BSD implicou, concretamente, o fim de ciclos de desenvolvimento colaborativos. Em outras palavras, a licença de Berkeley não garantiu – ao contrário da General Public Licence, que seria desenvolvida anos depois – que todas as modificações feitas sobre o *software* licenciado permanecessem "públicas", abrindo caminho para a criação de obras derivadas proprietárias. Essa característica da Licença BSD determinou que os diferentes projetos derivados do Unix se fragmentassem, uma vez que deu origem a projetos comerciais que passaram a concorrer fortemente entre si.

Nesse registro, a AT&T manteve a sua distribuição do Unix, mas com uma política de licenciamento alterada a partir de 1983, quando a distribuição do *software* passou a ser cobrada. A Universidade de Berkeley, por sua vez, continuou com a sua distribuição própria até 1989, quando em um esforço coordenado pela universidade, a comunidade de usuários do BSD-Unix reescreveu, por engenharia reversa, partes inteiras do programa que era, então, disputado pela AT&T. Esse projeto resultou no *Networking Release 1* – uma nova distribuição do Unix.[12] Paralelamente, inúmeras empresas – como, por exemplo, a HP, a IBM, a Sun Microsystem, entre outras – começaram a desenvolver versões próprias do sistema. Desse modo, no início dos anos de 1990, o Unix era distribuído em inúmeras versões concorrentes e, muitas vezes, incompatíveis entre si.

A ruptura das relações de colaboração entre Berkeley e a AT&T e o surgimento de versões proprietárias do Unix explicam-se, sobretudo, como dissemos, pelo fortalecimento do mercado de *software*, impulsionado pelo surgimento do computador pessoal. Essa mudança, associada a outras como o sucesso comercial da Microsoft, marca a transição da produção de *software* de um regime *público/científico* – caracterizado, entre outras coisas, pelo compartilhamento de informações, em especial do código-fonte do *software* – para um regime *privado/empresarial* – no qual a venda do *software* e a proteção do código-fonte, via propriedade intelectual, desempenham papel central.

A reação da comunidade *hacker*

O projeto GNU

Em 1971, o Massachusetts Institute of Technology (MIT) contratou Richard Stallman, então um jovem estudante de Harvard, como programador do seu Laboratório de Inteligência Artificial. Stallman tinha trabalhado no laboratório científico da IBM, em Nova York, após concluir o secundário e, em Harvard, perseguia seus interesses em Biologia e Física enquanto trabalhava no laboratório do MIT. Durante esses anos, tomou contato com a assim chamada "cultura *hacker*", uma subcultura muito particular, forjada pelos primeiros programadores do MIT nos anos de 1960 e que se baseava

em valores tais como a defesa da democratização do acesso aos computadores, a crença de que a informação deveria circular livremente, a desconfiança da autoridade baseada em credenciais, a defesa da meritocracia praticamente comprovada e a descentralização organizacional (Levy, 1984). Alguns desses valores, como a colaboração e a publicidade da informação, eram muito próximos do que era, então, chamado de *ethos* científico e, provavelmente, tinham sido incorporados à cultura *hacker* a partir da prática científica e universitária do MIT e de outras instituições de ensino e pesquisa. Nas palavras de Stallman:

> Quando comecei a trabalhar do Laboratório de Inteligência Artificial do MIT, em 1971, integrei uma comunidade de compartilhamento de *software* que já existia há muitos anos. O compartilhamento de *software* não era particularmente restrito à nossa comunidade [no MIT]; ele nasceu com os computadores, assim como o compartilhamento de receitas nasceu com o ato de cozinhar (Stallman, 1999).

Essa "cultura *hacker*", que em termos de valores gerais podia ser encontrada também em outras instituições como Stanford e Berkeley, sofreu um grande impacto a partir da emergência de um mercado autônomo de *software*.

Stallman costuma aludir a um evento que lhe serviu como uma espécie de "revelação" das mudanças que afetava a cultura dos programadores quando o *software* passou a ser visto como uma atividade comercial que deveria ser exercida de forma concorrencial e, portanto, produzido num regime outro que não o público/científico, que marcara o desenvolvimento do *software* até ali. O evento é o "mítico" episódio no qual a Xerox se nega a fornecer o código-fonte do *software* da impressora laser 9700 para Stallman. Em 1980, a impressora que era utilizada no laboratório do MIT começou a apresentar problemas no gerenciamento de trabalhos pela rede e Stallman buscou, sem sucesso, o código-fonte do *software* para poder consertá-lo, como já havia feito antes, no registro colaborativo que marcava, então, a pesquisa na universidade.

> Foi devido a essa filosofia do dar e receber que, quando Stallman se deparou com o defeito na impressora a laser da Xerox, ele não entrou em pânico. Ele simplesmente procurou uma forma de atualizar o conserto antigo [que tinha desenvolvido para um modelo anterior] ou então *hackear* o novo sistema. Ao procurar o *software* da impressora, no entanto, Stallman fez uma descoberta perturbadora. A impressora não tinha um *software*, pelo menos não um que ele ou outro programador pudessem ler (Williams, 2002).

Quando as empresas começaram a comercializar e, consequentemente, proteger os seus *softwares*, elas deixaram de distribuir o código de programação aos usuários, disponibilizando apenas o código binário que era lido e executado pela máquina. Assim, quem comprava o programa não tinha mais acesso ao código no qual ele havia sido programado (o código-fonte), o que dificultava que o programa fosse imitado por empresas concorrentes. O efeito colateral desta estratégia de proteger o patrimônio intelectual das empresas por meio do sigilo era que o *software* não podia mais ser desenvolvido – e incrementado – colaborativamente, já que as companhias estavam competindo entre si.

Nos anos seguintes, esse processo de comercialização intensificou-se ainda mais e o laboratório de Inteligência Artificial do MIT viu seus melhores programadores serem contratados pelo mercado, em particular por uma nova empresa chamada Symbolics. Stallman se viu no difícil dilema de aceitar o processo inexorável de competição comercial, indo trabalhar para alguma empresa, ou abandonar a computação. Foi em meio a essa dúvida em relação à sua trajetória que Stallman teve a ideia ousada de desenvolver um sistema operacional inteiro baseado nos valores de colaboração dos primeiros *hackers*. Seu projeto era criar um "mundo paralelo" de liberdade e colaboração que estivesse na contramão das práticas competitivas das empresas. Ele chamou esse projeto de GNU, um acrônimo recursivo cujo significado era "GNU Não é Unix", já que o sistema teria algumas características do sistema operacional Unix, mas não seria Unix.[13]

Em setembro de 1983, ele lançou um chamado à comunidade *hacker* pedindo a colaboração e, em 1985, lançou um manifesto que apresentava a filosofia do projeto:

> Eu considero uma regra sagrada a exigência de que eu compartilhe os programas de que gosto com outras pessoas que também gostem deles. Vendedores de *software* querem dividir os usuários para conquistá-los, fazendo com que cada usuário não compartilhe com os outros. Eu me recuso a romper a solidariedade com os outros usuários desta maneira. Eu não posso, em sã consciência, assinar um contrato com cláusula de sigilo ou uma licença de *software.* [...] Assim, para continuar a utilizar computadores sem desonra, eu decidi elaborar um conjunto de *softwares* livres, de forma que eu consiga utilizá-los sem qualquer recurso a *software* não livre. Eu me demiti do Laboratório de Inteligência Artificial do MIT para não permitir que o MIT tenha qualquer pretexto legal para impedir que eu distribua o GNU (Stallman, 2002, s/p).

O projeto GNU fazia um apelo moral para resgatar o "ato fundamental de amizade entre os programadores", que consistia no "compartilhamento de *software*". As novas práticas comerciais das empresas, que enfatizavam o sigilo e a competição, tornavam "fora da lei" a ação moral da amizade e da colaboração comunitária e era preciso, portanto, resgatá-la, conciliando a "hospitalidade" e a "obediência da lei" (*Idem*, s/p). A forma pela qual se efetivava essa conciliação do princípio moral e da lei era uma licença de *software* "pública", que foi chamada de Licença Pública Geral (General Public License ou *GPL)*. A GPL foi desenvolvida para licenciar os primeiros *softwares livres* que estavam substituindo os *softwares* proprietários. Era uma licença que subvertia o propósito original do direito autoral, de restringir a cópia para garantir a remuneração do detentor do direito. Como o titular original do direito autoral é "soberano" para estabelecer as condições de uso de suas obras, a GPL simplesmente afirmava que, no exercício desse direito, o detentor autorizava o livre uso, a modificação e a cópia do programa na forma original ou modificada, *desde que* essa cópia mantivesse as liberdades originais:

> Os contratos de licença da maioria das empresas de *software* tentam manter os usuários à mercê dessas empresas. Contra isso, nossa Licença Pública Geral tem o objetivo de garantir sua liberdade de compartilhar e modificar *software* livre – garantir que o *software* seja livre para todos os usuários. [...] Em particular, a Licença Pública Geral foi concebida para garantir que você tenha a liberdade de dar ou vender cópias de *software* livre, que receba o código-fonte ou tenha acesso a ele se quiser, que possa modificar o *software* ou usar partes dele em novos programas livres e que você saiba que pode fazer essas coisas. Para proteger seus direitos, precisamos criar restrições que proíbem alguém de negar esses direitos ou de pedir que você entregue os direitos (*Idem*, s/p).

O licenciamento da GPL diferia de estratégias anteriores de autorizar cópias e modificações (por exemplo, colocando as obras em domínio público) por exigir que versões subsequentes mantivessem as liberdades de modificar e copiar. Sem esse caráter "viral", o código liberado poderia ser reapropriado por empresas e fragmentado num processo competitivo subsequente, como tinha acontecido com a BSD. Stallman chamou esse efeito viral de *copyleft*, incorporando um trocadilho que havia sido sugerido por um amigo – Don Hopkins – em 1984 ou 1985.[14] O *copyleft* – que é a grande inovação conceitual da licença GPL – buscava evitar as apropriações de código que Stallman havia testemunhado, entre elas, uma experiência própria relativa a uma versão de um editor de textos que ele havia originalmente desenvolvido no MIT em 1976. O editor, chamado Emacs, tinha sido elaborado para o sistema operacional ITS (utilizado no laboratório do MIT) e depois fora reescrito para funcionar no Unix por James Gosling em 1981. Gosling tinha distribuído o código do programa livremente buscando colaboração, mas depois o vendeu para uma empresa chamada UniPress que impediu a sua distribuição *livre* a partir de então.

> No verão daquele ano, cerca de dois anos atrás [1984], um amigo me disse que devido à sua colaboração no desenvolvimento inicial do Gosling Emacs, tinha recebido permissão do Gosling, numa mensagem, para distribuir a sua versão do programa. Gosling tinha elaborado seu Emacs e distribuído de forma livre, conseguindo que muita gente colaborasse com ele no desenvolvimento, na expectativa, segundo o próprio Gosling diz no manual, de que seguiria com o mesmo espírito com o qual eu havia lançado o Emacs original. Então, ele deu uma facada pelas costas em todo mundo, colocando direitos autorais no programa, fazendo as pessoas prometerem não distribuí-lo e vendendo o *software* para uma empresa. [...] Neste ponto, a companhia para a qual Gosling tinha vendido o programa contestou o direito de meu amigo distribuí-lo e a mensagem [onde Gosling o autorizava a fazê-lo] estava armazenada em fitas de *backup* que ele não conseguia encontrar. E Gosling negou que tivesse dado permissão a ele (Stallman, 1986).

Com o *copyleft*, no entanto, esse tipo de manobra proprietária não era mais possível. A licença GPL garantia em termos legais sólidos que a obra podia ser modificada e redistribuída, e o caráter viral do *copyleft* fazia com que cópias ou obras derivadas adotassem compulsoriamente a mesma licença, difundindo as liberdades presentes na licença original. Foi sem dúvida devido ao caráter viral dessa forma de licenciamento que o projeto GNU conseguiu se expandir rapidamente nos anos seguintes. Com programas de excelente qualidade técnica, quase sempre distribuídos gratuitamente e acompanhados do código-fonte, muitos programadores, tanto do meio acadêmico como empresarial, passaram a fazer uso desse patrimônio "público" para produzir novos programas, mais complexos. E ao construir ferramentas baseadas em outras ferramentas licenciadas sob a GPL, terminavam inserindo essas obras derivadas no fundo público comum compulsoriamente. O crescimento do *software* livre assumia, assim, uma velocidade exponencial.

O Linux

No final dos anos de 1980, o projeto de criar programas "livres" que substituíssem os "proprietários" constituindo, assim, um sistema operacional completo estava próximo do fim. Faltava apenas um *kernel*, a parte central do sistema operacional encarregada da comunicação entre o *software* e o *hardware*. Enquanto a Free Software Foundation de Stallman se dedicava a um complexo projeto de *kernel* chamado Hurd, um estudante de pós-graduação da Finlândia, Linus Torvalds, anunciava o desenvolvimento de um *kernel* simples, mas eficiente, inspirado em um *kernel* acadêmico chamado Minix, que havia sido desenvolvido por Andrew Tanenbaum na Universidade de Vrije de Amsterdã. Em 25 de agosto de 1991, Torvalds enviou uma mensagem para o grupo de Usenet comp.os.minix anunciando o projeto do Linux e, no mês seguinte, lançou a primeira versão do *kernel* (versão 0.01) com a seguinte nota de *copyright*:

> Este *kernel* é ©1991 Linus Torvalds, mas ele pode, no todo ou em partes, ser redistribuído desde que você faça o seguinte:
> – Todo o código deve ser disponibilizado (livremente), se não com a distribuição, pelo menos quando requisitado
> – Notas de *copyright* devem permanecer intactas (na verdade, se você distribuir apenas parte dele, você precisará adicionar *copyright*, uma vez que não há ©s em todos os arquivos). Trechos menores podem ser copiados sem dar atenção a *copyrights*.
> – Você não pode distribuir exigindo uma taxa, nem mesmo para cobrir custos de "envio".
> Mande-me um *email* se você tiver perguntas.
> Infelizmente, um *kernel* sozinho não faz nada. Para ter um sistema que seja operacional, você precisará de um *shell*, compiladores, uma biblioteca etc. Tratam-se de partes separadas e que podem estar [licenciadas] sob *copyrights* mais estritos (ou frouxos). A maior parte das ferramentas utilizadas com o Linux são programas GNU e estão sob o *copyleft* GNU. Essas ferramentas não estão na distribuição. Para

> maiores informações, me escreva (ou escreva para o GNU) (Torvalds, s/d [a]).

Curiosamente, nessa primeira versão do Linux, a licença não é livre, pelo menos não na definição que Stallman havia dado ao termo. Desde o começo, Stallman havia incluído entre as liberdades do *software*, a de vendê-lo. O que tornava um *software* livre era o fato de que ele permitia a colaboração entre seus programadores, garantindo o acesso ao código-fonte e a livre reprodução e execução; e a mercantilização do *software* em si, por outro lado, não se constituía em um problema, embora, naqueles primeiros anos, se acreditasse que a oferta do código-fonte tornava inviável a venda de *software* livre – o que, depois, se mostrou possível, agregando-se a ele uma marca, por exemplo (Young, 1999). Era exatamente por isso que Stallman sempre enfatizou que o termo *free* da expressão *free software* referia-se à liberdade e não ao preço – na fórmula que ficou consagrada: "To understand the concept you should think of free as in free speech, not as free beer" (Stallman, 1996).

Torvalds afirma que a adoção de uma licença que permitia a cópia e o acesso ao código-fonte era parte da tradição acadêmica de compartilhamento de resultados científicos. Por outro lado, a restrição ao uso comercial vinha do temor de que a introdução do Linux no circuito comercial tirasse do autor o controle sobre o seu produto e desvirtuasse, consequentemente, seus objetivos técnicos:

> Quando, originalmente, publiquei o Linux, senti que estava seguindo as pegadas de séculos de cientistas e outros acadêmicos que construíram seu trabalho sobre as fundações de outros – nos ombros de gigantes, nas palavras de *Sir* Isaac Newton. [...] Creio que teria abordado a coisa de modo diferente se não tivesse sido criado na Finlândia, onde qualquer um exibindo o menor sinal de ganância é visto com suspeita, ou inveja. [...] E sim, eu sem dúvida teria abordado de maneira diferente a coisa do "sem dinheiro" se não tivesse sido criado sob a influência de um avô resolutamente acadêmico e um pai resolutamente comunista. De qualquer modo, eu não queria vender o Linux. E eu não queria perder o controle, o que significa que eu não queria que ninguém o vendesse também. Eu deixei isso claro na política de *copyright* que incluí na cópia da primeira versão que "subi" em setembro. [...] Pense bem. Você coloca seis meses da sua vida nessa coisa e quer torná-la disponível e quer tirar algo dela, mas não quer que as pessoas se aproveitem. [...] Fazia sentido para mim que a melhor maneira do Linux se desenvolver tecnologicamente era mantê-lo puro. Se houvesse dinheiro envolvido, as coisas ficariam obscuras. Se não há dinheiro em jogo, não há pessoas gananciosas (Torvalds e Diamond, 2001, pp. 93-94).

Está claro que, neste relato, Torvalds demonstra uma diferença em relação a Stallman quanto ao papel da permissão de uso comercial, referindo-se, no fundo, a *free* como em *free beer* ("Você não pode distribuir cobrando uma taxa"). Em um momento posterior, no entanto, devido ao fato de querer utilizar uma ferramenta licenciada em GPL no Linux (o compilador GCC, desenvolvido por Stallman), ele é obrigado a adotar a licença GPL, que não restringe o uso comercial (Torvalds, 1999). Essa adoção, aliás, mostra de maneira clara o efeito prático de uma licença viral, com *copyleft*. Na versão seguinte do Linux (0.0.12) de fevereiro de 1992, já aparece a seguinte nota:

> O *copyright* do Linux vai mudar: recebi algumas solicitações para torná-lo compatível com o *copyleft* do GNU, removendo a condição "você não pode distribuí-lo por dinheiro". Eu concordo. Proponho que o *copyright* seja mudado de forma a conformar-se com o GNU – se tiver a aprovação das pessoas que ajudaram a escrever o código. Presumo que não será problema para ninguém: se você tem queixas ("Eu escrevi o código presumindo que o *copyright* permaneceria do mesmo modo") me envie uma mensagem. Do contrário, o *copyleft* do GNU terá efeito a partir de primeiro de fevereiro. Se você não conhece a essência do *copyright* do GNU – leia (Torvalds, s/d [b]).

Assim, no começo dos anos de 1990, o projeto GNU havia realizado seu objetivo principal de de-

senvolver um sistema operacional completo baseado em uma licença livre (a versão 1.0 do Linux foi lançada em 1994). Nos anos seguintes, o Linux ganhou reputação como um sistema eficiente e estável; por suas virtudes técnicas e cooptação de esforços por meio do *copyleft*, despertou o interesse e a preocupação da comunidade empresarial.

A incorporação do regime público/científico e os novos modelos de negócios

As primeiras empresas de *software* livre, como a Cygnus e a VA Research, eram relativamente pequenas e baseavam-se no único modelo de negócios viável que se imaginava naqueles primeiros anos da década de 1990: a distribuição gratuita do *software* e a venda de serviços. No manifesto GNU, de 1985, Stallman já havia sugerido que os programadores poderiam garantir seus rendimentos deslocando a sua fonte de renda do licenciamento dos direitos autorais para os serviços de instalação, personalização e manutenção dos programas:

> A restrição de cópia não é a única base para os negócios no *software*. É a base mais comum porque é a que traz mais dinheiro. Se fosse proibida ou rejeitada pelo cliente, as empresas de *software* teriam que encontrar outras bases de organização que agora são pouco exploradas. Existem diversas formas de organizar qualquer tipo de negócio. [...] Um fabricante que introduza um novo computador poderá pagar pela transposição de sistemas operacionais para o novo *hardware*. A venda da atividade de ensino, orientação e serviços de manutenção também podem empregar programadores. Pessoas com novas ideias poderão distribuir programas como *freeware*, pedindo doações de usuários satisfeitos ou vendendo serviços de orientação. Já conheci pessoas que estão trabalhando desta maneira com sucesso. Usuários com necessidades parecidas poderão formar grupos de usuários e pagar taxas. Um grupo poderá contratar empresas de programação para escrever programas que os membros queiram utilizar (Stallman, 2002, s/p).

Mas o mundo do *software* livre só sentiu, de fato, o impacto do mundo dos negócios quando as grandes empresas de tecnologia da informação começaram a se aproximar daquele modelo de produção – mais ou menos no mesmo momento em que as primeiras empresas de *software* livre começam um processo de fusão (VA Research unindo-se à empresa Linux Hardware, e a Cygnus à Red Hat) e de abertura de capital na bolsa. O primeiro precedente significativo do interesse da parte tradicional do mundo dos negócios pelo *software* livre foi a decisão da Netscape de liberar o código do seu navegador para impedir a dominação do mercado pela Microsoft, migrando seu modelo de negócios totalmente para a oferta de serviços para o setor empresarial.

Com o crescimento da Microsoft no mercado de navegadores, a Netscape temia que ela passasse a impor padrões de protocolos não públicos, sabotando, assim, as suas concorrentes no mercado de servidores (que teriam dificuldade para servir páginas num padrão privado). À medida que ia dominando o mercado de navegadores, com o Internet Explorer, a Microsoft passava a utilizar uma codificação própria para páginas Web que apenas os seus próprios servidores eram capazes de produzir. Dessa maneira, ela transpunha a dominação do mercado de navegadores para o mercado de servidores. Para impedir, então, que a Microsoft dominasse o mercado de navegadores, a Netscape decidiu transferir o desenvolvimento do seu programa de navegação para a comunidade de *software* livre e, tendo garantido a concorrência neste mercado, passou a se concentrar nos servidores, que lhe eram mais rentáveis. Assim, em janeiro de 1998, a Netscape anunciou que passaria a distribuir o código do seu programa para a comunidade de *software* livre sob uma licença "herdeira" da GPL, passando a se concentrar no mercado de "soluções empresariais":

> A Netscape migrou com sucesso seu modelo de negócios no ano passado em direção à venda de *softwares* empresariais, para rendimentos ligados ao seu site, afastando-se assim dos rendimentos advindos de clientes isolados. No terceiro quadrimestre de 1997, os rendimentos advindos destes clientes representavam aproxi-

> madamente 18% da receita da Netscape – todo o resto vinha do *software* empresarial, serviços e o site. [...] Juntamente com o seu cliente gratuito [*free*], a Netscape anunciou também que está lançando um conjunto de produtos e serviços aperfeiçoados que alavancam seu cliente, tornando fácil para consumidores individuais e empresariais adotarem soluções Netscape. Os novos produtos e serviços reforçam a estratégia da Netscape de alavancar a penetração de mercado do seu popular cliente e seu concorrido site de Internet para fomentar ainda mais as vendas de soluções de *software* Netscape nos mercados doméstico e empresarial (Netscape, 1998).

A decisão da empresa tinha partido de funcionários que já colaboravam com a comunidade de *software* livre e que tinham sido muito influenciados por uma palestra – depois transformada em um influente artigo, chamado a "A catedral e o bazar" – proferida pelo programador de *software* livre, Eric Raymond, numa conferência em 1997. Nessa palestra, Raymond descrevia as virtudes técnicas da organização do trabalho no *software* livre, enfatizando a descentralização das responsabilidades, a autoiniciativa e o processo aberto de revisão por pares que permitia uma rápida identificação e correção de erros (Raymond, 2001a). Essa explicação do sucesso do *software* livre em criar ferramentas tão eficientes foi imediatamente tornada referência e influenciou muitas decisões posteriores de empresas tradicionais que aderiram ao *software* livre.

O interesse da Netscape por este tipo de *software livre* foi visto como uma janela de oportunidade para atrair outras empresas para o projeto do *software* livre. Muitos programadores acreditavam que o projeto do *software* livre, embora um grande sucesso técnico, era prejudicado por uma postura purista, confrontativa e desnecessariamente política do seu fundador original, Richard Stallman. Tal postura afastava as empresas e comprometia o crescimento do projeto. Assim, dez dias após o anúncio da Netscape, Eric Raymond, que havia sido contratado como consultor pela empresa para auxiliar na transição no seu modelo de negócios, reúne em Palo Alto, Califórnia, alguns dos principais líderes da comunidade de *software* livre (como Chris Peterson, John "Maddog" Hall e Michael Tiemann) para traçar estratégias para esse novo cenário de interesse das corporações. Linus Torvalds participou da reunião por vídeo-conferência. Stallman não foi convidado. Na reunião, os participantes decidem pela mudança do termo *free software* para o termo *open source software* (*software* de código aberto), que consideram menos político e ambíguo (pois *free* em *free software* dava a entender tanto que o *software* era livre como grátis) e por uma mudança no discurso da comunidade, da defesa moral da liberdade (de acesso ao código) para uma defesa pragmática da superioridade técnica do modo de produção do *software* (abordagem que tinha defendido no seu famoso artigo "A catedral e o bazar"). Cinco dias depois, Raymond faz um anúncio público à comunidade, propondo a substituição do termo "*software* livre" pelo termo "*software* de código aberto":

> O problema [com o termo "*software* livre"] é duplo. Em primeiro lugar, é confuso. O termo livre [*free*] é muito ambíguo (algo com o que a própria propaganda da Free *Software* Foundation tem que lidar constantemente). Será que *free* quer dizer "sem custos"? Ou será que significa "livre para ser modificado por qualquer um", ou mesmo uma outra coisa? Em segundo lugar, o termo deixa muitos empresários nervosos. Embora isso não me incomode nem um pouco, nós temos agora um interesse pragmático em converter essas pessoas, em vez de enfiar o dedo nos seus narizes. Existe agora a oportunidade de conseguir grandes ganhos no mundo empresarial sem colocar em risco os nossos ideais e nosso compromisso com a excelência técnica – então é hora de mudar de estratégia. Precisamos de um rótulo novo e mais adequado. [...] Devemos explicar publicamente os motivos para essa mudança. Linus Torvalds disse no evento World Domination 101 que a cultura do código aberto precisa fazer um esforço sério para se apropriar dos Desktops e abraçar o mundo dos negócios. Claro que ele está certo – e essa mudança de rótulo é parte do processo. Ele diz que estamos dispostos a trabalhar com o mercado e cooptá-lo para os

nossos próprios propósitos, em vez de permanecermos parados numa posição marginal e confrontativa (Raymond, 1998).

Com a adesão da Netscape ao *software* livre e a posterior aproximação de outras grandes empresas – como a IBM –, a Microsoft, que dominava o mercado de *softwares* no modelo tradicional, começou a mostrar preocupação. Em agosto de 1998, ela encomendou um memorando confidencial para o seu gerente de programas, Vinod Valloppillil, para avaliar as ameaças levantadas pelo crescimento do *software* de "código aberto" à liderança de mercado da empresa. O relatório, que vazou no final de outubro daquele ano (e, devido à data de vazamento, ficou conhecido como "Halloween papers"), mostrava a atenção que a Microsoft dava ao fenômeno do *software* livre e à potencialidade que via no método de produção:

> O *software* de código aberto é um processo que promove a rápida criação e preparação de características adicionais e correção de falhas numa base de código/conhecimento já existente. Recentemente, paralelamente ao crescimento da Internet, projetos de *software* de código aberto adquiriram profundidade e complexidade tradicionalmente associadas com projetos comerciais como Sistemas Operacionais e servidores para uso em situações críticas. Assim, o *software* de código aberto representa uma ameaça direta, de curto prazo, aos rendimentos e à plataforma da Microsoft – particularmente no espaço de servidores. Além disso, o paralelismo intrínseco e a livre troca de ideias no *software* de código aberto têm benefícios que não são replicáveis no nosso modelo de licenciamento atual e representam assim, no longo prazo, uma ameaça à repercussão junto aos programadores (Valloppillil, 1998, s/p).

Com a aproximação de grandes empresas como a Netscape e a IBM e a preocupação documentada da maior empresa do setor, os defensores do *software* de código aberto acreditavam que precisavam divulgar o seu potencial comercial e promover os novos modelos de negócios que deslocavam a fonte de rendimentos da venda de *software* para serviços e produtos agregados. Em fevereiro de 1998, um mês após o anúncio da Netscape, os ativistas do código aberto já haviam fundado uma organização (a Open Source Initiative, presidida por Raymond) para promover os novos modelos de negócio e, no ano seguinte, o mesmo Raymond publica outro artigo influente, dessa vez sistematizando os modelos de negócio existentes – baseando-se na experiência da Netscape e também no que faziam as empresas originalmente ligadas ao *software* livre como a Red Hat, a VA Linux, a Cygnus Solutions e a O'Reilly & Associates. Os modelos de negócios que descreviam eram principalmente quatro:

> *Posicionador de mercado:* Neste modelo, utiliza-se o *software* de código aberto para criar ou manter uma posição de mercado para um *software* proprietário que gera a receita direta. Na sua variante mais comum, um *software* cliente de código aberto proporciona a venda de *software* para servidores ou rendimentos de assinatura ou publicidade associados com um portal. A Netscape Communications Inc. estava seguindo esta estratégia quando abriu o código do navegador Mozilla no começo de 1998. […] *Enfeite de bugigangas:* Este modelo é para fabricantes de *hardware*. Pressões de mercado forçaram companhias de *hardware* a escrever e manter *software* (desde *drivers* para dispositivos, passando por ferramentas de configuração, até sistemas computacionais inteiros), mas o *software* em si não é o centro do lucro. É uma despesa indireta, normalmente substancial. Nesta situação, abrir o código é uma opção autoevidente. […] *Dê a receita, abra um restaurante:* Neste modelo abre-se o código de um *software* para criar uma posição de mercado, não para um *software* fechado (como no caso do posicionador de mercado), mas para serviços. Este modelo foi primeiro explorado pela Cygnus Solutions, talvez a primeira empresa de *software* de código aberto (1989). […] *Acessórios:* Neste modelo, vende-se acessórios para *software* de código aberto. Numa ponta, vende-se canecas e camisetas; noutra, documentação profissionalmente editada e produzida. A

O'Reilly & Associates Inc., editora de muitos excelentes volumes de referência sobre *software* de código aberto, é um bom exemplo de uma empresa de acessórios (Raymond, 2001b).

Sistematizando a relativamente restrita experiência do *software* livre de então, Raymond pôde apresentar para o mercado, de forma organizada, os modelos bem-sucedidos que haviam sido estabelecidos pelas empresas que haviam explorado as possibilidades de negócio com o *software* livre nos anos de 1990. A acuidade desta percepção e descrição podem ser medidas pelo fato de que, ainda hoje, esses são os quatro principais modelos de negócio de um setor produtor de *software* livre, cujo valor de mercado foi estimado em 2006 em 12 bilhões de euros (Ghosh, 2006) e no qual se envolvem empresas como a IBM, a Nokia e a HP.

O regime público/científico de produção do conhecimento

Como dissemos, uma das hipóteses que estrutura o presente trabalho é a de que a produção do *software* livre, ou de código aberto, se liga, por raízes históricas, ao regime de produção do conhecimento típico da ciência/tecnologia[15] produzida sob o regime público – cujo *locus* privilegiado são instituições como universidades e laboratórios governamentais – em oposição ao conhecimento produzido sob o regime privado – principalmente, embora não de forma exclusiva, nos laboratórios empresarias e departamentos privados de P&D. Até aqui, viemos tratando o regime público/científico de produção do conhecimento apenas em termos abstratos, sem definir suas características constitutivas. A partir de agora, faremos um esforço no sentido de caracterizá-lo a partir das dimensões que mais importam para a constituição da comunidade de *software* livre e, ao mesmo tempo, que determinaram a sua maior eficiência: a forma de organização/gestão do trabalho, o regime de recompensa/motivação subjetiva e a forma de gestão da propriedade intelectual. Essa reconstrução inspira-se, de um lado, nos estudos de sociologia da ciência (em especial, os que buscaram caracterizar a estrutura organizacional da ciência e suas práticas constitutivas: Merton, 1957, 1963; Bourdieu, 1975 e 2004; Shinn,1980, 2000a; 2000b; 2008a); e, de outro, nos estudos sobre a estrutura organizacional da comunidade *hacker* ou de *software* livre (Raymond, 1999, 2000, 2001a, 2001b; DiBona *et al.*, 1999; Bonaccorsi e Rossi, 2003, entre outros).

Embora os estudos clássicos de sociologia da ciência não enfatizem as práticas científicas da perspectiva da gestão do trabalho, inúmeros estudos o fizeram posteriormente. Desde as primeiras pesquisas sobre gestão do trabalho de cientistas em grandes laboratórios (por exemplo, Pelz e Andrews, 1966; Glaser, 1964; Kaplan, 1965) até estudos mais recentes sobre gestão da inovação e eficiência da produção e difusão do conhecimento (Dasgupta e David, 1994; Mazzoleni e Nelson, 1998; Nelson, 2004; Owen-Smith, 2001; Fleming e Sorenson, 2001, 2004) ressaltam as vantagens inerentes aos aspectos organizacionais da ciência para a gestão do trabalho em pesquisa e desenvolvimento.[16]

Nessa perspectiva, uma primeira dimensão essencial que concerne à organização do trabalho no regime público/científico é o *regime de excelência* que, em alguma medida, organiza a formação de hierarquias nesse âmbito.

Esse regime tem vários aspectos; o primeiro é a valorização, institucional e informal, da meritocracia. Esta é pensada, nesse caso, como um regime de controle do trabalho e hierarquização profissional baseado no reconhecimento do "mérito" que encarna, de forma mais ou menos precisa, a competência em uma determinada área. O mérito funciona, nesse caso, portanto, como um mecanismo específico de formação de "lideranças", as assim chamadas "lideranças científicas ou acadêmicas". Dito de outro modo, o controle de uma área de pesquisa – ou de um projeto de desenvolvimento de *software*, para aproximarmo-nos do nosso objeto – é exercido por aquele, ou aqueles, que têm mais legitimidade de fazê-lo porque são *reconhecidos*, pelos outros pesquisadores/desenvolvedores, como tendo mais mérito/competência para tanto. É evidente que o funcionamento da meritocracia, tal como a descrevemos aqui, tem uma dimensão *idealizada*, uma vez que, na prática, sobretudo a partir do fortalecimento da institucionalização da avaliação meri-

tocrática – a cristalização do "mérito" em títulos e índices de produtividade e eficiência acadêmica –, o controle do processo de controle da produção do conhecimento e a consequente formação de lideranças científicas tornou-se muito mais complexo.

De toda a forma, como na comunidade de *software* livre a institucionalização do mérito praticamente inexiste,[17] é mais fácil propor a existência de formas espontâneas de formação de lideranças que, sabemos, são centrais para a compreensão do funcionamento dos projetos de desenvolvimento.

Um segundo aspecto fundamental do regime de excelência científico/público é a *revisão por pares*, ou seja, um trabalho é considerado "correto" ou "válido", do ponto de vista científico, apenas quando os outros cientistas assim o reconhecem, o que pressupõe uma revisão minuciosa de cada trabalho. Mas a revisão por pares não avalia apenas "validade ou correção" de uma teoria mas, também, a sua qualidade e a sua legitimidade em relação a outras pesquisas.

A revisão por pares é um dispositivo institucional da ciência que está estritamente ligado, de um lado, à dimensão pública da atividade – é a ampla divulgação de resultados que permite que os cientistas tenham livre acesso às pesquisas e possam, a partir disso, revisar procedimentos e conclusões – e, de outro, ao seu caráter meritocrático: o reconhecimento dos "melhores" em uma determinada área depende diretamente da avaliação que os cientistas fazem do trabalho uns dos outros.[18]

Para alguns, o processo social de reconhecimento da confiança/qualidade das pesquisas (via revisão por pares) é um dos motivos fundamentais para se criticar o caráter racional da ciência; para outros, ao contrário, é justamente o que garante a racionalidade científica.[19]

Mas não é só a "racionalidade" da ciência que depende da revisão por pares – que pressupõe, nunca é demais lembrar, a ampla divulgação dos resultados e procedimentos de pesquisa – mas, também, o caráter eficiente e cumulativo da ciência: o fato de as pesquisas serem amplamente divulgadas impede a duplicação de esforços desnecessários e faz com que o conteúdo das pesquisas científicas seja acumulado ao longo da história (Nelson, 2004, p. 456).[20]

No caso do desenvolvimento de *software*, o fato de cada linha de código ser amplamente divulgada para a comunidade, que tem, assim, a possibilidade de as revisar em busca de erros e inconsistências – um procedimento, no geral, idêntico ao da revisão por pares –, permite não só que a comunidade reconheça seus melhores programadores – que, na maioria das vezes, controlam "espontaneamente" os projetos mais importantes – mas, também, que essa atividade se torne consideravelmente mais eficiente: nenhum erro fica muito tempo sem correção, nenhum problema evidente fica muito tempo sem solução e, sobretudo, ninguém reescreve uma linha que já tenha sido escrita por outra pessoa, ou seja, o caráter cumulativo da atividade impede que esforços sejam despendidos desnecessariamente.

Ainda do ponto de vista da organização do trabalho, outra dimensão que merece destaque no regime público/científico de produção do conhecimento é o caráter autogestionado do trabalho individual. Do ponto de vista *ideal*, os cientistas que trabalham no âmbito público têm maior liberdade na escolha dos problemas e das tarefas mais relevantes e mais espaço para determinação sobre quais destes melhor se adaptam às suas habilidades e competências. Além disso, em alguma medida, os cientistas escolhem de forma mais ou menos livre como empregar o próprio tempo, tanto no sentido de quanto e quando se dedicar como de quais tarefas e problemas precisam de mais tempo e quais devem ser encaradas primeiro. Assim, não são incomuns as histórias de cientistas que consomem madrugadas em torno de um mesmo experimento, ou dias e dias em torno de um único problema empírico ou teórico.

No caso do desenvolvimento de *software*, essas histórias também não são incomuns. O próprio desenvolvimento do Unix configura-se como um exemplo esclarecedor do quanto a autogestão do próprio tempo de trabalho pode ser produtiva nesses casos: Thompson, empregado do Bell Lab, desenvolveu o *kernel* do Unix durante suas férias de verão de 1969, quando tinha pleno controle do seu tempo, tanto assim que trabalhou em ritmo intenso para produzir um sistema operacional automático e multiusuário em aproximadamente quatro semanas.

Esta questão – da dedicação intensa e do controle do próprio tempo – remete imediatamente à outra dimensão essencial do regime público/científico de produção do conhecimento: a motivação subjetiva. O sistema de recompensas da ciência funciona de forma consideravelmente distinta do suposto sistema econômico/liberal de maximização de lucros. A motivação subjetiva dos que se dedicam à busca do conhecimento, considerando-se que essa ação se desenrola dentro de uma teia de relações sociais que configuram a estrutura institucional da ciência, não é financeira, mas de outra natureza. Os sociólogos da ciência empenharam-se sistematicamente na tentativa de entendimento do funcionamento desse sistema de recompensas (Merton, 1957; Bourdieu, 1975, 2004; Hagstrom, 1965) e, no geral, o descrevem como um sistema no qual a motivação é socialmente construída e intrinsecamente dupla: os cientistas buscam, ao mesmo tempo, o reconhecimento social do seu trabalho (por parte dos seus pares) e uma contribuição para o conhecimento de um determinado problema ou questão. Bourdieu descreve essa "ambiguidade estrutural da ciência nos seguintes termos:

> Há uma espécie de ambiguidade estrutural do campo científico (e do capital simbólico) que poderia ser o princípio objetivo da "ambivalência dos cientistas", já evocada por Merton, a propósito das reivindicações de prioridade: a instituição que valoriza a prioridade (ou seja, a apropriação simbólica), valoriza também o desinteresse e a "dedicação desinteressada ao avanço do conhecimento". O campo impõe, simultaneamente, a competição "egoísta" [...] e o "desprendimento" (Bourdieu, 2004, p. 77).

Bourdieu deixa explícita a indissociabilidade entre a *competição* e a *colaboração* na dinâmica de funcionamento da ciência: os cientistas colaboram entre si – compartilhando os resultados de suas pesquisas – e, ao mesmo tempo, o fazem para competir por reconhecimento, fonte de poder no campo. Essa dualidade, típica das trocas simbólicas, fez com que a dinâmica das trocas científicas pudesse ser explicada, também, pela teoria da dádiva, que atribui à "doação" esse caráter intrinsecamente ambíguo, em que a generosidade é inseparável do interesse (Hagstrom, 1972).

É interessante notar que essa dualidade inerente à ciência se expressa em um mesmo "momento": o da publicação dos resultados de pesquisa. A publicação é, simultaneamente, um ato de generosidade (o cientista *doa* o seu conhecimento para os outros) e um ato de disputa por mérito e por reconhecimento (o cientista *espera*, por meio dela, obter reconhecimento/prestígio/poder entre os seus pares).

O mesmo ocorre no processo de desenvolvimento de *software* no regime público/científico, ou seja, no *software* livre:

> Emergindo da universidade e do ambiente de pesquisa, o movimento [de *software* livre] adotou a motivação da pesquisa científica, transferindo-a para a produção de tecnologia que tem um potencial valor comercial. Compartilhar resultados permite ao pesquisador melhorar seus resultados de pesquisa a partir da resposta de outros membros da comunidade científica, ganhar o reconhecimento e, por conseguinte, o prestígio por seu trabalho. A mesma coisa acontece quando o código é compartilhado (Bonaccorsi e Rossi, 2003, p. 1245).

Assim, chegamos à última, e talvez mais importante, dimensão do regime científico/público de produção do conhecimento: o imperativo de publicação dos resultados. Para isso, a *propriedade intelectual* assume lugar central. Existindo nos dois regimes (no público/científico e no privado/empresarial) a propriedade intelectual assume sentidos completamente diferentes nos dois casos.

No regime privado/empresarial, em que o imperativo da competição entre firmas impõe a dinâmica do segredo ao processo de inovação e produção do conhecimento, a propriedade intelectual torna-se um dispositivo que restringe a reprodução e a utilização do conhecimento. Já no regime público/científico, em que o imperativo da publicação é o pilar sobre o qual se estruturam todas as práticas sociais de produção do conhecimento (a meritocracia, a liderança espontânea, a revisão aberta por pares e a motivação subjetiva), a propriedade intelectual

é mobilizada para autorizar a cópia, a utilização e a divulgação, acelerando o processo de inovação.

Essa relação entre abertura da ciência e aceleração da inovação foi destacada por autores como Richard Nelson, para quem "manter a ciência aberta é a política mais efetiva para garantir que o público extraia, dela, benefícios práticos" (Nelson, 2004, p. 456). Da mesma forma, Sorenson e Fleeming destacam que a norma da publicação é o que explica a influência positiva da ciência sobre as taxas de inovação tecnológica: "Mais do que acumular conhecimento para o ganho privado, a norma do comunismo [dos resultados], também conhecida como abertura, compele as descobertas a se disseminarem para outros [...] o que aumenta a eficiência das atividades científicas" (Fleeming e Sorenson, 2004, p. 1616).

Apesar da apologia desses e outros autores à eficiência inerente à produção do conhecimento sob o regime público/científico, a descrição que eles fazem, e a que aqui fizemos, do funcionamento da ciência sob o regime público condiz muito pouco com a realidade atual. É consenso de que práticas científicas vêm passando por profundas alterações que, em geral, as aproximam do regime privado/empresarial de produção do conhecimento. Segundo Shinn e Lamy:

> Na busca por novas formas de financiamento, e submetidas à pressão das demandas econômicas e sociais, as instituições científicas evoluem para modelos mais próximos da indústria. Elas se mercantilizam e tendem a submeter-se aos interesses comerciais e a inscrever-se numa lógica de oferta econômica que às vezes substitui, às vezes se mistura à lógica de oferta científica (Shinn e Lamy, 2006a, p. 23).

O incentivo ao patenteamento dos resultados de pesquisas realizadas nas instituições públicas e a política de direitos autorais das principais publicações científicas da atualidade alteram radicalmente a vigência do imperativo da ampla divulgação dos resultados (Milissard, Gingras e Gemme, 2003). Por outro lado, a orientação de pesquisas via financiamentos específicos, a diminuição crescente dos prazos e a cobrança crescente por produtividade individual configuram-se como novos dispositivos de gestão do trabalho que subvertem o caráter autogestionado da atividade científica, aproximando o funcionamento da universidade ao de empresas modernas (Yates, 2000; Shinn e Lamy, 2006a e b; Kleinman e Vallas, 2001, 2008).

Mas mesmo em meio a tantas mudanças profundas e inegáveis, alguns pesquisadores ainda ressaltam aspectos de continuidade e permanência (Shinn e Lamy. 2006a e b; Godin e Gingras, 2000). Além disso, apesar dessas mudanças que vêm afetando as práticas sociais que constituem a ciência nos seus aspectos mais fundamentais, o regime público/científico de produção do conhecimento parece ser, em muitos casos, muito mais eficiente do que o regime privado/empresarial, o que, no caso do desenvolvimento de programas de computador, fez com que ele se impusesse como regime dominante, inclusive entre as grandes empresas de *software*.

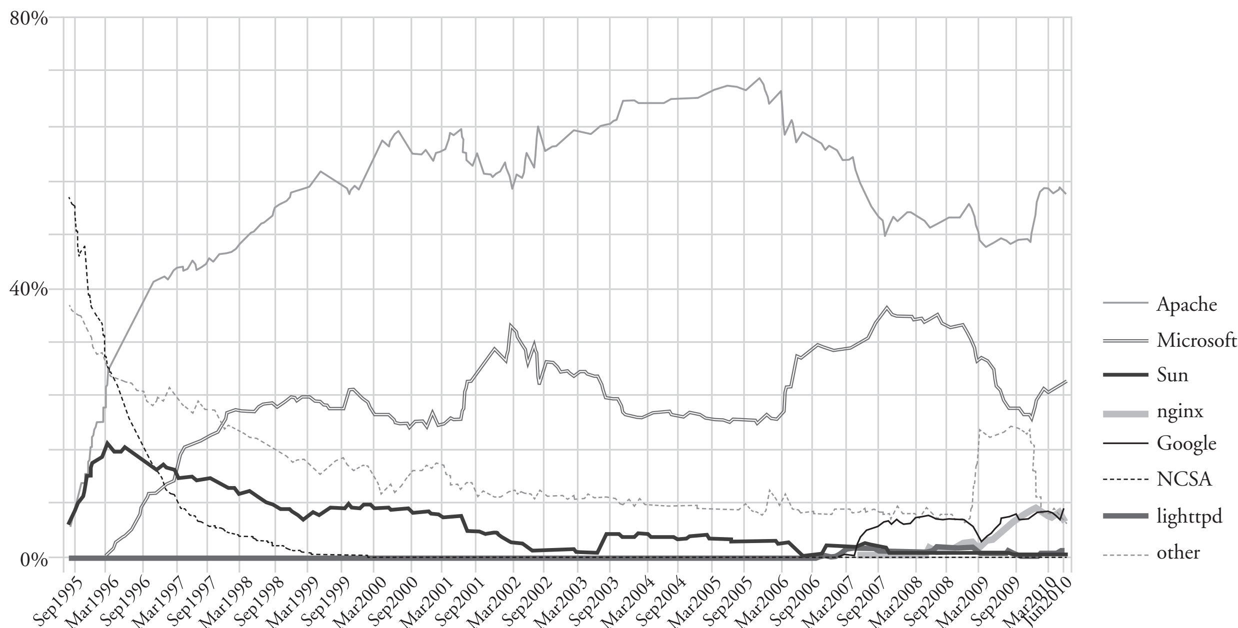
Produtos mercantis e extramercantis

A disputa por consumidores entre o *software* produzido no regime público e o produzido no regime privado implica uma competição *sui generis* entre um produto mercantil e outro extramercantil. Mas em que terreno ocorre essa disputa? No mercado? Como pode um produto sem valor mercantil disputar mercado com um produto mercantil? Quando o servidor Apache da Fundação Apache compete com o Internet Information Services da Microsoft e ganha mais consumidores, o que temos? Uma "conquista de mercado" pelo Apache?

A história do Apache é, de todas aquelas do *software* livre, talvez a mais significativa para entendermos a interação entre comunidade e mercado. O Apache domina "o mercado de servidores" praticamente desde o começo da Web, desde o primeiro semestre de 1996, para sermos precisos. O projeto Apache nasceu para coordenar correções e melhoramentos feitos voluntariamente por programadores e *webmasters* para o primeiro servidor popular, o "http daemon" (httpd), desenvolvido de forma "livre"[21] por Robert McCool, da Universidade de Illinois. Quando McCool e outros programadores-chave do httpd foram contratados pela Netscape para desenvolver seu servidor em 1994, diversos

Gráfico 1

Market Share de Servidores Web



Fonte: Netcraft, 2008.
Versão colorida disponível em <http://news.netcraft.com/archives/2010/06/16/june-2010-web-server-survey.html>.

outros que utilizavam o *software* continuaram desenvolvendo-o individualmente, adicionando funções e corrigindo falhas. O projeto Apache nasce, em fevereiro de 1995, como um consórcio de oito destes programadores (que se chamam de "grupo Apache") para coordenar os esforços de desenvolvimento do httpd por meio da Internet. Em apenas um ano, o *software* reunindo as melhorias sobre o código do httpd já é o servidor Web mais utilizado no mundo e permanece assim até hoje.

Embora tenha sido, desde o começo, um sucesso na disputa "por mercado", o Apache foi inicialmente concebido como um "esforço voluntário, completamente financiado por seus membros e não por vendas comerciais" e que buscava permanecer como um servidor "de domínio público". Ele tinha sido desenvolvido "para resolver questões de provedores www e programadores de httpd de tempo parcial relativas ao mau comportamento do httpd"[22]. Era, portanto, um empreendimento extraprofissional de alguns programadores, embora diretamente ligado às suas atividades profissionais. Um dos fundadores do grupo Apache descreveria, em um artigo em coautoria em 1997, a motivação dos programadores do projeto dessa maneira:

> Cada voluntário do grupo Apache tem (pelo menos) um emprego "de verdade", normalmente relacionado com serviços de Web ou pesquisa sobre protocolos. Eles colaboraram na produção e suporte ao servidor Apache apenas por autointeresse esclarecido: ao consorciar seus esforços, o produto resultante é muito mais funcional e robusto do que o que poderiam ter produzido individualmente (Fielding e Kaiser, 1997, s/p).

Esse depoimento indica a natureza da colaboração no desenvolvimento do *software*. De um lado, temos os pesquisadores da universidade, para quem

a colaboração é norma profissional; de outro, os *webmasters* e programadores que competem na provisão de serviços para empresas (elaboração e manutenção de páginas Web). Como o *software* do servidor é apenas um meio para a hospedagem do site, que é o verdadeiro produto final, os programadores de páginas Web podem colaborar no *software* do servidor, o que facilita o trabalho de todos, sem dar vantagem competitiva para nenhum dos programadores de site. Além disso, o desenvolvimento de uma plataforma comum garante que os padrões da Web sejam universais (pois as páginas serão servidas da mesma forma), sem o domínio de uma empresa que os privatize, garantindo os fundamentos da competição de mercado. A Fundação Apache explica:

> Nós acreditamos que as ferramentas de publicação *online* devem estar nas mãos de todos e que as empresas de *software* deveriam extrair dividendos da provisão de serviços adicionais tais como módulos especializados e suporte, entre outras coisas. Nós sabemos que normalmente se vê como uma vantagem econômica para uma empresa o "domínio" do mercado – na indústria do *software*, isso significa controle estrito de um canal, de maneira que os outros precisam pagar para o seu uso. Isso é feito tipicamente pela "posse" de protocolos pelos quais empresas conduzem negócios às custas de todas as outras empresas. Na medida em que os protocolos da World Wide Web permaneçam sem a "posse" de uma única empresa, a Web permanecerá um campo onde poderão jogar empresas pequenas e grandes. Por isso, a "propriedade" dos protocolos precisa ser evitada.[23]

Mas, o que significou, do ponto de vista econômico, o domínio do Apache no mercado de servidores? Como o Apache é uma ferramenta livre, disponível gratuitamente na Web, não se pode falar, rigorosamente, que ela dominou o mercado, já que é um produto não mercantil. Podemos, ao contrário, dizer que a oferta desse produto reduziu o mercado de servidores. Em outras palavras, a oferta de um *software* concorrente não mercantil, de qualidade igual ou superior aos produtos mercantis, impediu que o mercado de *software* para servidores se estabelecesse e se expandisse em toda a sua potencialidade. Embora concorressem por consumidores, alguns produtos estavam no mercado, enquanto outros estavam fora. E cada vez que o produto extramercantil ganhava um consumidor, *stricto sensu*, ele reduzia o mercado. O *software* livre pode ser visto, então, como uma barreira à ampliação do mercado de venda de *software* e, se levarmos em conta as motivações históricas de Stallman e dos primeiros programadores de *software* livre, essa barreira à expansão do mercado é uma reação à natureza anticolaborativa do trabalho de produção de *software* no regime privado.

Se a tese de Eric Raymond e dos outros ativistas do *software* de código aberto estiver correta, a vitória do Apache na disputa com o Internet Informations Service da Microsoft e outros concorrentes menores não se deve apenas à sua gratuidade, mas também à qualidade técnica que é fruto da maneira como o *software* é produzido. O regime público/científico, traria as seguintes vantagens:

- Organizaria o trabalho de maneira mais eficiente, permitindo que os próprios programadores determinem onde melhor podem alocar suas competências e permitindo também que a comunidade identifique "espontaneamente" as lideranças, de maneira estritamente meritocrática, dispensando as credenciais hierárquicas (como títulos e cargos).
- Permitiria uma evolução técnica mais acentuada, impedindo que o desenvolvimento de uma mesma funcionalidade fosse duplicado em firmas concorrentes. Duas iniciativas potencialmente concorrentes que se unam num projeto de *software* de código aberto passam a acelerar o processo de desenvolvimento.
- Permitiria a identificação mais rápida de falhas e novas demandas, permitindo que o usuário relate e, se tiver competência, resolva o problema ou proponha melhorias. Seria um processo de monitoramento aberto da estrutura e da funcionalidade do programa que geraria correções mais eficientes e inovações mais frequentes.

Entendemos que, se essas características foram, de fato, determinantes para o sucesso competitivo

do *software* livre, elas devem ter forçado uma mudança nas indústrias tradicionais que tiveram que adotar alguns desses procedimentos para concorrer de forma eficiente com as alternativas não mercantis, em um processo que poderia ser descrito em termos "evolucionários".[24] Se a forma de produção do *software* de código aberto mostrou-se mais eficiente do que a produção de *software* proprietário e reduziu em mais da metade o mercado de venda de *software* para servidores, então as empresas atuando na esfera mercantil precisam, para continuar competindo, adotar as práticas mais eficientes do *software* de código aberto e, consequentemente, migrar sua fonte de recursos para serviços e produtos associados – sejam *softwares* proprietários associados ao *software* livre ou serviços de instalação, personalização e manutenção. Em outras palavras, a eficiência da alternativa não mercantil pode ter obrigado as empresas que não conseguem mais competir a colaborar nessa esfera não mercantil e concentrar a competição em serviços e produtos associados ao *software* como, por exemplo, manutenção e customização.

Isso talvez explique um levantamento recente sobre os vínculos empregatícios dos programadores do *kernel* do Linux (núcleo do sistema operacional), indicando que apenas 18,2% deles não têm vínculos empregatícios com empresas que contribuem com o *kernel* – assim, supostamente, não contribuem para o *kernel* na qualidade de empregados de empresas (Kroah-Hartman *et al.*, 2009). Embora os diferentes levantamentos já feitos sobre a profissionalização dos programadores de *software* livre não tenham metodologias e universos uniformes, o que dificulta a comparação, é possível fazer algumas inferências sobre tendências gerais. Sabemos que nos anos de 1980 praticamente não havia programadores pagos dedicados a colaborar com o projeto GNU e que, em 2002, no levantamento feito por Ghosh, conhecido como Floss Survey, apenas 16% dos programadores entrevistados eram pagos para colaborar com desenvolvimento, e uma parte não determinada desses 16% era paga por entidades não mercantis, como universidades ou fundações (Ghosh, 2002). De forma que é bastante provável que a participação de programadores pagos por empresas para colaborar em projetos de *software* livre tenha sido originalmente muito baixa ou nula e tenha aumentado, com um salto no período de "mercantilização" que vai de 1997 até 1999 e esteja crescendo gradualmente a partir de então, até a altíssima concentração atual, quando mais de 80% dos colaboradores são assalariados em projetos de grande interesse mercantil como o *kernel* do Linux ou o servidor Apache.

Se nossa hipótese estiver correta, então podemos tirar algumas conclusões e propor especulações sobre a subsistência de um regime público de produção no setor de *software* inserido no contexto mais geral de uma sociedade de mercado. Parece-nos que a eficiência do regime de produção público/científico forçou, por um processo competitivo – de natureza *sui generis*, porque não se tratava de uma entidade com natureza propriamente econômica –, as empresas tradicionais a migrarem seu modelo de negócios para serviços e produtos associados e a colaborarem na produção do *software*. Assim, a eficiência do regime de produção público "protegeu" o mercado de *software* para servidores da mercantilização, mas apenas ao custo de forçar as empresas do setor a supermercantilizarem os serviços para subsidiar a desmercantilização do *software*.

Para aqueles que, como nós, entendem que a desmercantilização da produção de *software* é positiva porque impede a degradação do processo colaborativo do regime público/científico em competição e transformação da propriedade comum do *software* em propriedade privada – o que resulta em perda de qualidade e barreiras de preços –, talvez seja o caso de perguntar se não estaríamos presenciando um processo de transformação do próprio regime público de produção de *software* – suas práticas, valores e estrutura organizacional –, à medida que ele interage e, por vezes, se subordina ao regime empresarial. Embora a lógica do processo de competição tenha levado as empresas a colaborar na produção do *software*, transferindo a mercantilização e a competição econômica para o setor de serviços, percebemos algumas tendências notáveis de degradação do processo colaborativo de produção de *softwares*:

- Empresas de *software* livre têm utilizado diversas estratégias para não compartilhar programas que desenvolvem a partir de código com licenças *copyleft* (como a GPL). Essas estraté-

gias incluem disponibilizar o código-fonte de maneira pouco pública ou postergar a publicação, entre outras (Henkel, 2006).

- Empresas estão transferindo seus serviços da venda do *software* para a execução de serviços pela Internet. Como não distribuem o *software* (pois este é executado pela Internet a partir de um servidor), essas empresas não precisam, de acordo com os termos da licença GPL, compartilhar as mudanças que fizeram a partir de um código licenciado em *copyleft*. Na revisão dos termos da licença GPL, em 2007, discutiu-se a inclusão de uma cláusula para obrigar a distribuição do código também quando o *software* é acessado remotamente (um dispositivo que já aparece numa variante da GPL chamada Afero-GPL), mas nas negociações com a comunidade (da qual fazem parte as empresas interessadas), essa cláusula foi abandonada. Esse tipo de não compartilhamento de modificações de *softwares* livres que não são distribuídos, mas disponíveis pela Internet tende a crescer com o advento de serviços de "computação nas nuvens", como redes sociais, *webmails* e *sites* que disponibilizam *streamming* de vídeos pela Web. É por isso que Richard Stallman tem chamado a computação nas nuvens de uma "cilada" (Stallman, 2008).
- Embora a licença de *software* livre mais utilizada hoje ainda seja a GPL, a maior parte das empresas tem preferido licenças sem dispositivos virais como o *copyleft*, o que permite que lancem produtos proprietários simultaneamente aos produtos comunitários – de maneira que a colaboração é apenas uma forma de capturar desenvolvimento de código não remunerado. Se as licenças não virais começarem a prevalecer é possível que, a médio prazo, os códigos livres se fragmentem em produtos proprietários concorrentes, como aconteceu com o Unix.
- Como a forma de recompensa do regime privado – o assalariamento – tem prevalecido entre os grandes colaboradores dos principais projetos de *software* livre – em oposição a uma colaboração predominantemente voluntária no passado –, é possível que a contribuição não remunerada passe pouco a pouco a ser vista como ingenuidade ou anacronismo até desaparecer a médio prazo (Benkler, 2006). Se a colaboração subsistisse sem a dimensão voluntária das contribuições de indivíduos autônomos e pesquisadores, teríamos então um curioso caso de colaboração entre empresas, talvez análogo ao estabelecimento dos consórcios para a determinação de padrões técnicos.
- Por fim, é possível que as empresas comecem a utilizar estratégias mais abertas para conduzir os projetos colaborativos de *software* livre ou de código aberto de acordo com os seus interesses – hoje elas apenas contratam bons programadores que atuam "espontaneamente" como líderes mais ou menos formais dos projetos – e passem a utilizar estratégias para valorizar seus serviços que é onde está a sua fonte principal de faturamento. Isso poderia acontecer, por exemplo, na produção de pouca documentação ou documentação de baixa qualidade, de forma a tornar mais necessária a prestação de serviço. Observa-se ainda, de maneira mais nítida, na ênfase dada à produção de desenvolvimento para servidores corporativos em detrimento de desenvolvimento para os *desktops*, cujo mercado de serviços é ainda pouco desenvolvido na avaliação das empresas.[25]

Embora essas sejam tendências reais observáveis, há também muita resistência por parte dos programadores que valorizam sua forma tradicional de organização do trabalho. O que podemos vir a assistir, nos próximos anos, talvez não deva ser caracterizado propriamente como uma "degradação do regime público", mas como a constituição de um novo regime que misture características do regime público e privado e que, provavelmente, encontre uma contrapartida nas modificações que o regime público está sofrendo em universidades e institutos públicos de pesquisa que estão incorporando traços do regime de produção e organização de empresas (Kleinman e Vallas, 2001, 2008). Esse novo regime poderá fazer convergir a organização produtiva das universidades e das empresas – em particular das empresas de tecnologia ou daquelas que concentram pesquisa e desenvolvimento sem contrapartida industrial significativa –, tornando obsoletos, na sua forma

tradicional, os valores públicos da colaboração e do desinteresse, os quais precisarão ser retomados em novas bases. Mas esses desdobramentos estão, para os analistas, ainda no registro da especulação. O que talvez não seja mera especulação é a constatação de que, no encontro entre o regime público/científico e o regime privado/empresarial, as práticas colaborativas dos programadores de *software* podem estar adquirindo novos sentidos que, por vezes, contradizem seu significado original. Onde há continuação e expansão de práticas colaborativas, podem estar emergindo, na verdade, novos padrões de competição e mercantilização.

Notas

1 A literatura sobre o tema não usa, necessariamente, o termo regime, mas noções semelhantes que nos permitem traçar paralelos e associações. Henkel (2006), por exemplo, trabalha com a ideia de tipos de inovação (inovação clássica ou fechada em contraposição à inovação coletiva ou aberta). Gambardella e Hall (2006) falam em modos de pesquisa (pesquisa de domínio público em contraposição à pesquisa proprietária). Laat (2005) fala em regimes de propriedade (copyright X copyleft), Bonaccorsi e Rossi (2003), em processos de inovação; Osterloch e Rota (2007), em modelos de inovação. Todos eles, porém, aceitam a ideia de que existem duas "formas" de desenvolver software, as quais correspondem determinadas características.

2 Segundo Campbell-Kell e Aspray (1996), no começo da década de 1990 havia, no máximo, cem programadores de computador nos Estados Unidos.

3 Eram elas a North American Aviation, a Douglas Aircraft Company, a IBM, a Ramo-Woolridge, e a RAND Corporation.

4 Um sistema operacional é, basicamente, o conjunto de programas que fazem um aparelho eletrônico qualquer – como um computador, um celular etc. – funcionar. Em suma, é o software que viabiliza o funcionamento do Hardware. Ele se constitui, portanto, em um elemento fundamental do funcionamento de um computador.

5 "Para que a colaboração exista, precisa haver uma cultura de colaboração. Antes do PACT essa cultura simplesmente não existia" (Kim, 2005).

6 Um consent decree – em português, compromisso de cessação ou decreto de consentimento – é um acordo judicial feito pelas partes de um processo antes do julgamento judicial.

7 Segundo Weber, Thompson desenvolveu a primeira versão do Kernel do Unix durante as quatro semanas de suas férias de verão do Bell Labs em 1969. O Unix, nessa época, chamava-se ainda Unics, numa menção explícita ao Multics (Weber, 2004, p. 25).

8 IV Simpósio Sobre Princípios de Sistemas Operacionais.

9 Kernel (em português, cerne ou núcleo) é a parte mais importante de um sistema operacional, porque possibilita a comunicação dos componentes de hardware (a parte física da máquina) e o software (o sistema lógico ou programa).

10 A única exigência era que nos licenciamentos futuros e na divulgação do software fossem dados créditos à Universidade da Califórnia. Mesmo essa "cláusula de propaganda", como era chamada, foi retirada, posteriormente, das licenças BSD.

11 Disponível em <http://www.freebsd.org/copyright/license.html>.

12 O projeto Networking Release 1 foi o primeiro a envolver uma comunidade de programadores/usuários de software num projeto de desenvolvimento amplamente centrado em contribuições via Internet. Esse processo formalizou, pela primeira vez, um modelo de coordenação do trabalho de desenvolvimento de *software* que viria a se consolidar na década de 1990, qual seja, a de um desenvolvimento distribuído e coordenado pela rede, em que cada colaborador voluntário recebe um reconhecimento nominal quando da divulgação do projeto concluído.

13 Eric Raymond afirma que a escolha de criar um sistema compatível com Unix era reveladora de uma tradição mais antiga de colaboração: "A escolha de RMS [Richard Matthew Stallman] de modelar o sistema operacional GNU sobre o Unix reflete, num nível técnico, o fato cultural de uma comunidade [de colaboração] pré-existente" (Raymond, 2003).

14 Na verdade, a expressão já havia sido utilizada antes, em 1965, por Gregory Hill e Kerry Thornley em Principia Dischordia, obra na qual Hopkins provavelmente se inspirou. Ver Grassmuck (s.d).

15 A diferença entre ciência e tecnologia não será discutida em detalhe aqui, porque não é fundamental para o problema que apresentamos. Ver, nesse sentido, Cupani (2006) e Forman (2007).

16 Owen-Smith, em especial, destaca que – desde que a Segunda Guerra Mundial, quando a ciência deixou paulatinamente de ser vista como um instrumento de

capacitação bélica e passou a representar uma ferramenta para o aumento da competitividade da economia nacional, a atividade científica passou a ser vista como um trabalho; dessa perspectiva, ainda, as normas científicas permitem uma maior eficiência da gestão das atividades científicas (Owen-Smith, 2001, p. 427).

17 Segundo Raymond, “Existe uma meritocracia muito estrita (o melhor artesão ganha) e existe um forte ethos de que a qualidade poderia (na verdade, deveria) falar por si mesma” (2000, p. 12).

18 Esse processo tornou-se mais complexo no caso da ciência por conta tanto da fragmentação das práticas e tradições de pesquisa como da institucionalização da avaliação meritocrática, que favorece a adoção de critérios estritamente quantitativos, o que, às vezes, conflita com formas mais tradicionais de avaliação e hierarquização. No caso do software livre, no entanto, essa descrição ainda é bastante procedente.

19 Latour e Woolgar (1979), por exemplo, destacam o caráter de negociação e, portanto, de construção social por trás da consolidação de verdades científicas, o que coloca em xeque, de certa forma, o caráter estritamente racional da ciência. Já Bourdieu, discordando explicitamente dos primeiros, aponta na direção contrária. Para ele, “o fato de os produtores tenderem a ter como clientes apenas os seus adversários mais rigorosos, os mais competentes e críticos, portanto os mais inclinados e os mais aptos a validar a sua crítica, é, para mim, o ponto arquimediano em que nos podemos basear para explicar cientificamente a razão da científica” (Bourdieu, 2004, p. 78). Assim como Bourdieu, muitos entendem que é a publicação dos resultados que constitui a garantia da racionalidade científica.

20 Richard Nelson é um dos autores mais sensíveis à relação entre eficiência e abertura da ciência. Segundo ele: “Todos os cientistas são livres para testar os resultados dos seus colegas e achá-los válidos ou não, e para construir sobre esses resultados os seus próprios trabalhos. Porque os resultados das pesquisas científicas são deixados no domínio publico para teste e desenvolvimentos futuros, é que o conjunto do conhecimento científico aceito é confiável [...] e o conhecimento científico é cumulativo. Essas são as razões básicas do porquê de a empresa científica ter sido tão eficiente como engrenagem de descobertas” (2004, p. 456).

21 A licença do httpd não era livre nos termos da Free Software Foundation. Ela permitia a distribuição e a modificação do código, mas restringia a redistribuição comercial. Trata-se de uma opção semelhante àquela primeira adotada por Linus Torvalds – o que demonstra, aliás, como Stallman, por meio da GPL, modificou o entendimento intuitivo que se tinha de licenças “livres”. Ver <http://hoohoo.ncsa.uiuc.edu/docs/COPYRIGHT.html>.

22 Disponível em <http://web.archive.org/web/19961028122409/www.apache.org/docs/FAQ.html>.

23 Disponível em <http://httpd.apache.org/ABOUT_APACHE.html>.

24 “Os ambientes de mercado oferecem uma definição de sucesso para as firmas, e essa definição está muito próxima à habilidade delas de sobreviver e crescer. Padrões diferenciais de sobrevivência e crescimento numa população de firmas podem produzir mudanças nos agregados econômicos que caracterizam aquela população, ainda que as características correspondentes das firmas sejam constantes” (Nelson, 2005. p. 26).

25 É o caso, por exemplo do ex-programador do kernel do Linux, Con Kolivas, que ao sair da equipe de desenvolvimento afirmou que a baixa performance dos desktops deve-se ao fato de estarem repletos de “porcarias” empresariais (Kolivas, 2007).

BIBLIOGRAFIA

ARMER, Paul. (1973), “Entrevista concedida a Robina Mapstone”. Archives Center, National Museum of American History, 17 abr. Disponível em <http://invention.smithsonian.org/downloads/fa_cohc_tr_armer730417.pdf>.

BENKLER, Yochai. (2006), *The wealth of networks*. New Haven, Yale University Press.

BERNSTEIN, Morton. (1973), “Entrevista concedida a Robina Mapstone”. Archives Center, National Museum of American History, 14 mar. Disponível em <http://invention.smithsonian.org/downloads/fa_cohc_tr_bern730314.pdf>.

BIAGIOLI, Mario. (1998), “The instability of authorship: credit and responsibility in contemporary biomedicine”. *Journal of the Federation on American Societies for Experimental Biology*, 12: 3-15.

BONACCORSI, Andrea & ROSSI, Cristina. (2003), “Why open source *software* can succeed”. *Research Policy*, 32: 1243-1258.

BSD License. (1974-1994), “Berkeley Software Distribution License”. Disponível em <http://www.freebsd.org/copyright/license.html>.

BOURDIEU, Pierre. (1975), “La spécificité du champ scientifique”. *Sociologie et Societé*, 7: 91-118.

_________. (2004), *Para uma sociologia da ciência.* Lisboa, Edições 70.

CAMPBELL-KELL, Martin & ASPRAY, William. (1996), *Computer: a history of the information machine.* Nova York, Basic Books.

CUPANI, Alberto. (2006), “La peculiaridad del conocimiento tecnológico”. *Scientiae Studia*, 4 (3): 353-371.

DASGUPTA, Partha & DAVID, Paul. (1994), “Toward a new economics of science”. *Research Policy*, 23: 487-521.

DIBONA, Chris *et al.* (1999), *Open sources: voices from the open source revolution.* Sebastopol, O’Reilly.

FLEMING, Lee & SORENSON, Olav. (2001), “Technology as a complex adaptive system: evidence from patent data”. *Research Policy*, 30: 1019-1039.

_________. (2004), “Science and the diffusion of knowledge”. *Research Policy*, 33: 1615-1634.

FIELDING, Roy & KAISER, Gail. (1997), “The Apache HTTP Server Project”. *IEEE Internet Computing*, 1(4).

FORMAN, Paul. (2007), “The primacy of science in modernity, of the technology in the postmodernity, and of ideology in the history of technology”. *History and Technology*, 23 (1-2): 1-153.

GAMBARDELLA, Alfonso & HALL, Bronwyn. (2006), “Proprietary versus public domain licensing of software and research products”. *Research Policy*, 35: 875-892.

GATES, Bill. (1976), “An open letter to hobbyists”. *Homebrew Computer Club Newsletter*, 2 (1). Disponível em <http://en.wikipedia.org/wiki/Image:Bill_Gates_Letter_to_Hobbyists.jpg>.

GHOSH, Rishab. (2002), “Free/libre and open source software: survey and study”. Maastricht, International Institute of Infonomics.

_________. (2006), “Economic impact of open source software on innovation and the competitiveness of the Information and Communication Technologies (ICT) sector in the EU”. Ann Arbor, Merit.

GHOSH, Rishab & GLASER, Barney. (1964), *Organizational scientists: their professional careers.* Indianapolis, Bobbs-Merril.

GODIN, Benoît & GINGRAS, Yves. (2000), “The place of universities in the system of knowledge production”. *Research Policy*, 29: 273-278.

Grassmuck, V. (s/d), “Towards a new social contract: free-licensing into the knowledge commons”. Disponível em <http://waste.informatik.hu-berlin.de/Grassmuck/06_KNAW-Social_Contract.pdf>.

HAGSTROM, Warren. (1965), “Social control in science”, *in* _________, *The scientific community*. Nova York, Basic Books.

_________. (1972), “La diferenciación de las disciplinas”, *in* B. Barnes (org.), *Estudios sobre sociologia de la ciencia,* Madri, Allianza Editorial.

HENKEL, Joachim. (2006), “Selective revealing in open innovation processes: the case of embedded Linux”. *Research Policy*, 35: 953-969.

KAPLAN, Norman. (1965), “Professional scientists in industry: an essay review”. *Social Problems*, 13: 88-97.

KLEINMAN, Daniel & VALLAS, Peter. (2001), “Science, capitalism, and the rise of the ‘knowledge worker’: the changing structure of knowledge production in the United States”. *Theory and Society*, 30: 451-492.

_________. (2008), “Contradiction, convergence and the knowledge economy: the confluence of academic and industrial biotechnology”. *Socio-Economic Review*, 6 (2): 283-311.

KOLIVAS, Con. (2007), “Why I quit: kernel developer, Con Kolivas (entrevista)”. Disponível em <http://apcmag.com/why_i_quit_kernel_developer_con_kolivas.htm>.

KIM, Eric Eugene. (2005), “Everything is known: discovering patterns of emergent collaboration”, *in* Chris DiBona *et al*., *Open source 2.0.* Sebastopol, O’Reilly Media. Disponível em <http://blueoxen.com/paper/discovering-patterns/>,

KROAH-HARTMAN, G. *et al*. (2009), “Linux kernel development: how fast it is going, Who is doing it, what they are doing and Who is sponsoring it”. São Francisco, The Linux Foundation.

LAAT, Paul. (2005), "Copyrigth or copyleft? An analysis of property regimes for *software* development". *Research Policy*, 34: 1511-1532.

LATOUR, Bruno & WOOLGAR, Steve. (1979), *Laboratory life: the social construction of scientific facts.* Beverly Hills, Sage.

LEVY, Steven. (1984), *Hackers: heroes of the computer revolution.* Garden City, Doubleday.

LINUX WEEKLY NEWS. (s/d), "Who wrote 2.6.20?". Disponível em <http://lwn.net/Articles/222773/>.

MERTON, Robert. (1957), "Priorities in scientific discovery: a chapter in sociology of science". *American Sociological Review*, 22 (6): 635-659.

_________. (1963), "The ambivalence of scientists". *Bulletin of the John Hopkins Hospital*, 4: 237-282.

_________. (1973), *The sociology of science: theoretical and empirical investigation.* Chicago, University of Chicago Press.

MAZZOLENI, Roberto & NELSON, Richard. (1998), "The benefits and costs of strong patent protection: a contribution to current debate". *Research Policy*, 27: 273-284.

MILISSARD, Pierrick; GINGRAS, Yves & GEMME; Brigitte. (2003), "La commercialisation de la recherché". *Actes de la Recherche en Sciences Sociales*, 148: 57-57.

NELSON, Richard. (2004), "The market economy and the scientific commons". *Research Policy*, 33: 455-471.

NELSON, Richard & Siney Winter. (2005), *Uma teoria evolucionária da mudança econômica.* Campinas, Editora da Unicamp.

NETSCAPE. (1998), "Press release: Netscape announces plans to make next-generation communicator source code available free on the Net". Netscape.

OSTERLOH, Margit & ROTA, Sandra. (2007), "Open source software development: just another case of collective invention? *Research Policy*, 36: 157-171.

OWEN-SMITH, Jason. (2001), "Managing laboratory work through skepticism: processes of evaluation and control". *American Sociological Review*, 66: 427-452.

PELZ, Donald & ANDREWS, Frank M. (1966), *Scientists in organizations.* Nova York: John Wiley.

RAYMOND, Eric. (1998), "Goodbye, 'free software'; hello, 'open source'". Disponível em <http://www.catb.org/~esr/open-source.html>.

_________. (1999), "A brief history of hackerdom", *in* Chris Dibona *et al.*, *Open sources: voices from the open source revolution.* Sebastopol, O'Reilly. Disponível em <http://www.catb.org/%7Eesr/writings/cathedral-bazaar/hacker-history/index.html#id2763962>.

_________. (2000), "Homesteading the noosphere". Disponível em <http://catb.org/~esr/writings/cathedral-bazaar/homesteading/>.

_________. (2001a), "The cathedral and the bazaar", *in* _________, *The cathedral & the bazaar*, Sebastopol, O'Reilly. Disponível em <http://catb.org/~esr/writings/cathedral-bazaar/cathedral-bazaar/>.

_________. (2001b), "The magic cauldron", *in* _________, *The cathedral & the bazaar*, Sebastopol, O'Reilly. Disponível em <http://catb.org/~esr/writings/cathedral-bazaar/magic-cauldron/>.

_________. (2003), "A fan of freedom: thoughts on the biography of RMS". Disponível em <http://www.catb.org/~esr/writings/rms-bio.html>.

SALUS, Peter. (1994), *A Quarter Century of Unix.* Londres, Addison-Wesley.

SHINN, Terry. (1980), "Division du savoir et especificité organisationelle: les laboratoires de recherche industrielle em France". *Revue Française de Sociologie*, 21 (1): 3-35.

_________. (2000a), "Formes du travail scientifique et convergence intellectuelle: la recherche technico-instrumentale". *Revue Française de Sociologie*, 41 (3): 447-473.

_________. (2000b), "Axes temáthiques et marchés de diffusion". *Sociologie et Societés*, 32 (1): 43-69.

_________. (2002), "Nouvelle production du savoir et triple hélice: tendences du prêt-à-penser las sciences". *Actes de la Recherche en Sciences Sociales*, 141: 21-30.

_________. (2008a), "Regimes de produção e difusão de ciência: rumo a uma organização transversal do conhecimento". *Scientiae Studia*, 6 (1): 11-42.

_________. (2008b), "Desencantamento da mo-

dernidade e da pós-modernidade: diferenciação, fragmentação e a matriz de entrelaçamento". *Scientiae Studia*, 6 (1): 43-81.

SHINN, Terry & LAMY, Erwan. (2006a), "L'autonomie scientifique face à la mercatilisation". *Actes de la Recherche en Sciences Sociales*, 164: 22-49.

_________. (2006b), "Paths of commercial knowledge: forms and consequences in university--enterprise synergy in scientist-sponsored firms". *Research Policy*, 35: 1465-1476.

SHINN, Terry & RAGOUET, Pascal. (2005), *Controverses sur la science: pour une sociologie transversaliste de l'activité scientifique.* Paris, Raisons d'Agir.

_________. (2008a), *Controvérsias sobre a ciência: por uma sociologia transversalista da atividade científica*. São Paulo, Associação Filosófica Scientiae Studia/Editora 34.

_________. (2008b), "Regimes de produção e difusão de ciência: rumo a uma organização transversal do conhecimento. *Scientiae Studia*, 6 (1): 11-42.

STALLMAN, Richard. (1986), "Lecture at KTH". Disponível em <http://www.gnu.org/philosophy/stallman-kth.html>.

_________. (1996), "The free software definition". Disponível em <http://www.gnu.org/philosophy/free-sw.html>.

_________. (1999), "The GNU Operating System and the Free Software Movement", *in* C. DiBona *et al.*, *Open sources: voices from the open source revolution*, Sebastopol, O'Reilly. Disponível em <http://www.oreilly.com/catalog/opensources/book/stallman.html>.

_________. (2002), "The GNU Manifesto", *in* Richard Stallman (org.), *Free software, free society: selected essays of Richard Stallman*. Boston, Free Software Foundation. Disponível em <http://www.gnu.org/gnu/manifesto.html>.

_________. (2008), "Cloud computing is a trap, warns GNU founder Richard Stallman (entrevista)". Disponível em <*guardian.co.uk.*>, 29 set.

THOMPSON, Ken & RITCHIE, Dennis. (1973), "The Unix time sharing-system". Trabalho apresentado no IV Symposium on Operating Systems Principles. Disponível em <http://www.informatik.uni-trier.de/~ley/db/conf/sosp/sosp73.html>.

TORVALDS, Linus. (1999), "The Linux Edge", *in* C. DiBona *et al.*, *Open sources: voices from the open source revolution*, Sebastopol, O'Reilly. Disponível em <http://www.oreilly.com/catalog/opensources/book/linus.html>.

_________. (s/d [a]), "Notes for Linux release 0.01". Disponível em <http://www.kernel.org/pub/linux/kernel/Historic/old-versions/RELNOTES-0.01>.

_________. (s/d [b]), "Release notes for Linux v. 0.12". Disponível em <http://www.kernel.org/pub/linux/kernel/Historic/old-versions/RELNOTES-0.12>.

TORVALDS, Linus & DIAMOND, David. (2001), *Just for fun: the story of an accidental revolutionary*. Nova York, Harper Collins.

UNITED STATES DISTRICT COURT. (1956), "Consent Decree. Civil Action nº 72-344". Nova York. Disponível em <http://www.cptech.org/at/ibm/ibm1956cd.html>.

VALLOPPILLIL, Vinod. (1998), "Open source software: a (new?) development methodology". Disponível em <http://catb.org/~esr/halloween/halloween1.html>.

WEBER, Steven. (2004), *The success of open source.* Cambridge, Harvard University Press.

WILLIAM, Sam. (2002), "Free as in freedom: Richard Stallman's crusade for free software. Disponível em <http://oreilly.com/openbook/freedom/>.

YATES, Michael. (2000), "Us versus them: laboring in the academic factory". *Montly Review*, 51 (8): 40-49.

YOUNG, Robert. (1999), "Giving it away: how red hat Software stumbled across a new economic model and helped improve an industry", *in* Chris Dibona *et al.*, *Open sources: voices from the open source revolution*, Sebastopol, O'Reilly. Disponível em <http://www.oreilly.com/catalog/opensources/book/young.html>.

ACTIVIST-DRIVEN INNOVATION: UMA HISTÓRIA INTERPRETATIVA DO *SOFTWARE* LIVRE

Maria Caramez Carlotto e Pablo Ortellado

Palavras-chave: Inovação; Propriedade intelectual; *Software* livre; Regimes de produção de conhecimento.

A compreensão de que existem dois regimes distintos para a produção de *software* é cada vez mais comum na literatura sobre o tema. O que não é tão comum, e se configura, portanto, como a contribuição mais original deste artigo é, de um lado, a abordagem histórica da configuração desses dois regimes e, de outro, a análise dos fatores que determinam o sucesso técnico e "comercial" de um regime em detrimento do outro. Assim, trabalhamos com duas hipóteses complementares: a primeira, de que o desenvolvimento do *software* livre pertence historicamente ao regime público/científico de produção do conhecimento, ou seja, o *software* livre mimetiza a organização da comunidade científica porque tem nela as suas raízes históricas; e a segunda de que, em uma "competição de mercado", o regime público/científico se mostrou mais eficiente e, por isso, obrigou as empresas que trabalhavam no regime privado/empresarial a adotar o *software* livre ou de código aberto.

ACTIVIST-DRIVEN INNOVATION: AN INTERPRETIVE HISTORY OF FREE SOFTWARE

Maria Caramez Carlotto and Pablo Ortellado

Keywords: Innovation; Intellectual property; Free software; Knowledge production regimes.

The understanding that there are two distinct regimes for the production of software is increasingly common in literature. What is not so common, and is therefore the most original contribution of this paper is, on the one hand, the historical approach to the configuration of those regimes and, on the other hand, the analysis of the factors determining the technical and commercial success of one regime over the other. Furthermore, we have worked with two additional hypotheses: first, that the development of free software historically belongs to the public/scientific knowledge production regime – i.e., free software mimicking the organization of the scientific community because it has its historical roots in it; and secondly, that in a "market competition" environment the public and scientific regime has proven more efficient and has therefore forced companies working in the private/business regime to adopt free or open source software.

ACTIVIST-DRIVEN INNOVATION: UNE HISTOIRE INTERPRÉTATIVE DU LOGICIEL LIBRE

Maria Caramez Carlotto et Pablo Ortellado

Mots-clés: Innovation; Propriété intéllectuelle; Logiciel libre; Régimes de production du savoir.

La compréhension de l'existence de deux systèmes distincts pour la production de logiciels est de plus en plus courante dans la littérature sur le sujet. Ce qui n'est pas très usuel, et se configure, par conséquent, comme étant la contribution la plus originale de cet article est, d'une part, l'approche historique de la configuration de ces deux systèmes et, d'autre part, l'analyse des facteurs qui déterminent le succès technique et "commercial" d'un système au détriment d'un autre. Nous avons, ainsi, travaillé avec deux hypothèses complémentaires: la première, celle selon laquelle le développement de logiciels libres appartient historiquement au régime publique/scientifique de production du savoir, c'est-à-dire, le logiciel libre reproduit l'organisation de la communauté scientifique, car il incorpore des racines historiques de cette communauté; la seconde s'appuie sur le fait que dans une "concurrence du marché", le système public et scientifique s'est avéré plus efficace et a donc forcé les entreprises qui travaillaient sous le régime privé d'adopter le logiciel libre ou open source.